REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 July 2002	Final Report

4. TITLE AND SUBTITLE

Adaptive Finite Element Methods for Rotary Wing Aerodynamics

5. FUNDING NUMBERS

N68171-01-M-5866

6. AUTHOR(S)

Carlo L. Bottasso, Stefano Micheletti, Riccardo Sacco

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Politecnico di Milano, Milano, Italy

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES

Naval Regional Contracting Center, Detachment London, Block 2, Wing 11, DoE Complex, Eastcote Road, Ruislip, Middlesex, UK, HA4 8BS

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

R&D 8928-AN-01S

11. SUPPLEMENTARY NOTES

Conference Programme and Book of Abstracts (Preconference), First International Conference 17-20 June 2002, London, United Kingdom, 157 pages.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution authorized for public release.

12b. DISTRIBUTION CODE

A

ABSTRACT (Maximum 200 words)

We report on the research activity developed under the present contract in the field of anisotropic mesh adaption and error estimation. With respect to the former problem, we describe a method for the insertion of anisotropic mesh layers in general tetrahedral grids. For a-posteriori error estimation, we describe an implementation of a recovery-based estimator and its extension to anisotropic metric-based adaption strategies. Selected examples illustrate the characteristics of the proposed procedures. We conclude by commenting on the results of the research activity and by discussing possible future work in the same field.

14. SUBJECT TERMS

US Naval Research, Italy, Rotorcraft aerodynamics, Finite elements, adaptivity

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Adaptive Finite Element Methods for Rotary Wing Aerodynamics

Carlo L. Bottasso (PI), **Stefano Micheletti**, **Riccardo Sacco**

Politecnico di Milano, Milano, Italy

Contract Number **N68171-01-M-5866**

R&D 8928-AN-01S

Final Report

The Research reported in this document has been made possible through the support and sponsorship of the U.S. Government through its European Research Office of the U.S. Army. This report is intended only for the internal management use of the Contractor and U.S. Government.

CONTENTS

Abstract

We report on the research activity developed under the present contract in the field of anisotropic mesh adaption and error estimation. With respect to the former problem, we describe a method for the insertion of anisotropic mesh layers in general tetrahedral grids. For a-posteriori error estimation, we describe an implementation of a recovery-based estimator and its extention to anisotropic metric-based adaption strategies. Selected examples illustrate the characteristics of the proposed procedures. We conclude by commenting on the results of the research activity and by discussing possible future work in the same field.

Key Words

Finite elements, adaptivity, anisotropic meshes, unstructured tetrahedral and hybrid grids, a-posteriori error estimation, recovery-based error estimation, metric-based grid adaption.

1. INTRODUCTION

In this report we describe the work we have performed with the support of the U.S. Army Research Office through its European Office in London under a contract with the Politecnico di Milano. The work here described aims at investigating promising solution procedures for the anisotropic mesh adaptive analysis of PDE's, with particular reference to the compressible Euler and Navier-Stokes equations.

Indeed, solution features in many different branches of computational mechanics are associated with strong directionality. In the area of external fluid dynamics, for example, we can cite the case of rotor aerodynamics as a typical situation where the anisotropy of the solution plays a very important role and poses serious limitations to the effectiveness of isotropic mesh methods. For example, when refining shocks or pressure signals in high speed impulsive noise analysis, it would be extremely desirable to have the ability to generate elements that are highly stretched in the direction of the shock surface. In the tracking of vortices, gradients are much higher in the radial direction so that highly stretched elements in the direction of the vortex itself are highly desirable. On the contrary, isotropic mesh refinements are inefficient, in the sense that they tend to generate too many grid points, most of them bearing no practical utility. Since computer resources are always limited and since three dimensional problems are always extremely demanding in terms of computing time and memory, the lack of efficiency in this sense seriously undermines the ability of such isotropic processes to demonstrate grid independent results.

The work is here organized in two main sections. Section 2 discusses the mesh modification procedures that are used for inserting layers of anisotropic elements into three-dimensional grids. The methodologies here illustrated produce anisotropic grids of excellent quality by carefully avoiding the construction of elements with large dihedral angles. This result is achieved by inserting structured layers of elements that are finally tetrahedronized.

Next, Section 3 discusses the problem of error estimation for driving the adaption process. Two procedures are described. The first makes use of a generalization of the Zienkeiwicz-Zhu (ZZ) error estimator, that produces robust estimates at low computational cost. The method is essentially based on the computation of higher order information regarding the gradients of the solution. The estimate is then obtained by comparing the original computed gradient and the reconstructed one. This procedure is known to produce excellent results for a number of different applications, and it is here implemented in our research parallel adaptive CFD code. One drawback of this approach is that no anisotropic information is directly available from the computed estimator for driving a metric-based anisotropic adaption process. To circumvent this limitation, a second procedure is proposed that is based on estimates of the interpolation error which explicitely take into account the anisotropy of the solution and of the mesh elements. These estimates,

which in principle would be prohibitively expensive to evaluate for realistic problems, are then made computable by using the Zienkeiwicz-Zhu reconstructed gradients. This way a local metric can be constructed for driving an anisotropic iterative adaptive mesh method.

Finally, Section 4 presents some final remarks and comments on the work conducted so far. In particular, we mention the highlights and the present limitations of the layer insertion algorithms reported. We propose to complement the present procedures with metric-based point insertion algorithms, that, at the price of lower quality grids, will extend the applicability of the adaptive software to handle more general situations. Next, we briefly discuss the error estimation procedures and we propose plans of future work in this area. Finally, we conclude listing the scientific publications produced under the support of this contract.

2. Procedures for Anisotropic Mesh Adaption

2.1. **Introduction and Motivation.** Many problems in fluid and solid mechanics present solution features that are inherently directional. For example shock waves are characterized by gradients that are very high in one local direction, while substantially smaller in the other two. Similarly, a vortex shed by the tip of a wing or a helicopter blade presents gradients in the radial direction that are much higher than in the direction of its core. In all these and many other cases, anisotropic meshes are highly desirable for their evident economic advantages.

It appears that structured procedures are better suited at generating highly anisotropic grids. However, unstructured grids can discretize non-manifold domains of arbitrary topological complexity with very little user intervention and consequent limited work load. Furthermore, unstructured grids can be locally adapted in order to capture the relevant solution features to the required level of approximation. These two characteristics of the unstructured approach are contributing to its ever-increasing success in complex applications in many diverse disciplines.

In general, the introduction of anisotropic regions can be obtained during unstructured mesh generation with the use of local mappings [5], although the amount of element stretching is somewhat limited by the need to provide some form of smooth transition to isotropic regions. Furthermore, the method tends to create large dihedral angles that affect the overall quality of the resulting mesh.

One possible alternative is represented by the introduction of structured mesh regions in the areas of localization, while using an unstructured discretization to fill the rest of the computational domain. The nature of structured discretizations makes it possible for the contrasting requirements of very high elongation and good mesh quality, in particular control of large dihedral angles, to be more easily achieved. For example, ref. [28] develops a semi-structured two-dimensional procedure that first forms layers of quadrilateral elements in the regions of high localization and then triangulates the rest of the domain using an advancing front method. However, in three spatial dimensions the use of local structured hexahedral meshes would lead to non-conformity at the structured/unstructured interfaces, with obvious complications.

In three spatial dimensions, hybrid tetrahedral/prismatic grids have been used with success in regions of high gradients in proximity of model faces (boundary layers). The prismatic regions present a structured discretization in the direction of the solution gradients, which allows for extremely high elongations with good control of dihedral angles. At the same time, the grid furnishes an unstructured triangular discretization of the boundary layer face that allows for a conforming interface with the rest of the isotropic grid. Refs. [17, 23] develop a method for the generation of hybrid

tetrahedral/prismatic grids, where a prismatic mesh is obtained by marching away from the model boundary an initial triangular surface mesh. An octree-advancing front method is then used for filling the rest of the domain with tetrahedra.

The advancing layers method [21, 10], in its various flavors, uses similar ideas. The method starts with a triangulation of the model faces, that is then inflated along quasi-normal directions using a modified advancing front algorithm. Various forms of pruning and collapsing of elements in problem regions is used for improving the robustness and generality of the method. The process is terminated by filling the rest of the domain with an isotropic mesh obtained via a standard advancing front method. A generalization of the method based on multiple growth directions is detailed in ref. [15]. Since multiple growth curves can be used at any given point, gaps are created in the boundary layer mesh. Special blends and transition elements are implemented to hide the gaps generated by the multiple growth directions that expose highly anisotropic faces to the mesh generator. Both the advancing layers and the hybrid tetrahedral/prismatic method are designed primarily for generating boundary layer meshes; the anisotropic refinement of internal features such as shock waves can be performed by adding fictitious entities to the underlying solid model, for example by inserting an internal model face to represent the shock wave surface.

In this section of the report, we present a set of automated procedures for the directional refinement of three-dimensional unstructured grids defined in complex domains, with the specific goal of allowing large values of element stretching and good control of the mesh quality. The method is suitable both for the generation of boundary layer meshes and for the adaptive refinement of internal features with local one-dimensional anisotropy, i.e. with large gradients in one single local direction as in the typical case of shock waves. The proposed procedures start with an initial isotropic or anisotropic grid and insert layers of prisms in the regions of localization. The location of the regions that need to be adapted and the stretching of the inserted layers are assumed to be known from the use of an appropriate error estimator or from user-defined input.

For boundary layers, the procedure is normally driven by user-supplied input. Attributes specifying (possibly non-uniform) boundary layer thickness, number of layers and their geometric distribution are associated with faces in the solid model wherever a boundary layer needs to be grown. The meshing process starts by locally deflating an initial isotropic grid, obtained with any suitable mesh generator. The deflation of the mesh is obtained with a specially designed mesh motion algorithm based on force control whose goal is to make room for the boundary layer region. The highly stretched tetrahedra of the boundary layer region are then created by first inserting stacks of prisms in the resulting void, between the initial and the final positions of the external triangulation of the grid, and then by tetrahedronization of each prism. A simple algorithm ensures the determination of the correct prism

splitting template that needs to be used in each case in order to obtain a conforming grid.

For features within the problem domain (e.g. shocks), the tetrahedral mesh is first cut along the feature. This creates a triangulated surface within the mesh. Next, the two sides of the cut are marched away from each other in order to make room for the prismatic mesh. Once again, the required space is obtained by deformation of the tetrahedral grid.

In both cases, the procedures are based on the idea of opening a gap in the mesh that has the same triangular discretization on the two sides, and filling the resulting void with stacks of prisms. We refer to this procedure as "wedging". The meshing processes are based on a solid-model description of the domain which allows for a consistent handling of the model geometry and avoids precision losses in the presence of curved boundaries.

The proposed method achieves two main goals: first, it is able to introduce "structured", and therefore well-behaved, regions of elements within arbitrary unstructured grids; second, this result is obtained with a remarkably simple process. In fact, all advancing layers methods grow the boundary layer mesh in a void. This is a difficult task for highly anisotropic meshes, and requires a number of accompanying complex correction procedures. On the contrary, the proposed method makes careful use of the deformations imposed on an already exiting isotropic grid together with appropriate mesh smoothing processes in order to obtain a natural way of controlling the mesh quality by automatically adjusting the anisotropic layer thickness and the growth directions. Once calibrated, the procedures are remarkably simple, being essentially based on a robust mesh smoother and on a set of prism splitting templates, in addition to a few support routines. As far as the generation of boundary layers is concerned, the procedures here proposed are not as general as the advancing layers method of ref. [15], although they seem to offer a simpler alternative in many practical cases. Furthermore, they can also be applied to the refinement of internal features, which would not be easy to achieve with the advancing layers method.

This section is organized as follows. Some basic terminology used throughout this work is introduced in Section 2.2. Section 2.3 introduces the problem by discussing the internal insertion of anisotropic layers. The required mesh cutting is explained in 2.3.1. Section 2.4 gives a general overview of the boundary layer meshing algorithm, providing details on the prism stack creation and on the wedge tetrahedronization. Next, we discuss the mesh motion algorithm in Section 2.5, together with a line identification scheme used for ordering during smoothing. Details on the practical implementation of the gridding algorithm are given in Section 2.6, where we analyze the problem of visibility of the deflation directions, we comment on several heuristic quality enhancement procedures that we have found useful and effective, on the treatment of transition zones at the layer boundaries and on the reclassification of the mesh entities. The proposed anisotropic mesh

generation algorithm is tested on a number of practical examples in Section 2.7. The quality of the resulting meshes is analyzed using geometric criteria.

2.2. **Solid Modeling and Terminology.** Any given geometric domain can be described by a boundary representation (BRep) of its topology. The basic topological information is represented by topological entities that bound each other, in the order regions, faces, edges and vertices. An attribute information associated with each topological entity specifies its geometry, e.g. a surface associated with a face or a curve associated with an edge. Throughout this work we use the capital letters **R**, **F**, **E**, **V** to indicate regions, faces, edges and vertices, respectively, in a solid model.

Grids can also be represented using a boundary representation, although not all four entities are always necessarily present in any data structure. In fact, the sets of entities required for representing a grid is somewhat dependent on the application and the use that the application is making of the grid itself [4]. In any case, all topological entities in a mesh are generated by discretizing a parent entity in the geometric model. For example, a mesh face can be obtained either by discretizing a model face or by discretizing a model region. The term "classification" identifies this unique relationship between each entity in the mesh with an entity in the model. The symbols **r**, **f**, **e**, **v** are used to indicate regions, faces, edges and vertices, respectively, in a mesh.

The procedures described in this work make use of a mesh data structure that is interfaced with a computer aided design (CAD) system, that provides geometric and/or topological interrogations on the nature of the domain through a limited set of functional operations. Topological interrogations are used for ensuring the compatibility of the mesh with the model, while geometric interrogations are here used for the proper placement of mesh vertices on the model boundary.

2.3. **Meshing of Internal Features.** We start by describing the wedging algorithm for the case of internal features, which represents the most complex case. The case of boundary features will then just use a subset of the described procedures.

At this point of the discussion, we will simply assume that a directional error estimator is available for the problem at hand. A possible simple method of computing the regions of localization and the required local stretching is discussed in ref. [28]. In any case, the procedure should be able to locally define a minimum edge size h_{min} in the local direction of anisotropy **n** and a maximum edge size h_{max} in the orthogonal direction.

As a preliminary step, passes of edge-based refinement are applied to the mesh in order to meet the requirement on the local value of h_{max}. Next, the requirement on h_{min} needs to be enforced. To this end, we create a surface that gives a geometric approximation of the feature to be refined. The geometry is obtained through the use of the error estimator that indicates the grid points where high values of stretching are needed. The targeted

grid points are fitted on to a surface, that in this way approximates the mid-surface of the region of high gradients.

2.3.1. *Mesh Cutting and Local Optimization.* Next, the mesh is cut along the surface. Each tetrahedron intersected by the surface is split into tetrahedra according to the corresponding subdivision pattern. In turn, this provides a triangular conforming discretization of the surface. The resulting surface mesh is stored in a list for subsequent use in the wedging process.

In general, cutting the mesh along the localization surface will locally generate elements of poor quality that will negatively effect the final adapted grid. Furthermore, the quality of the surface mesh that will provide the triangular faces for the stack of prisms introduced during wedging, will be poor. Therefore, local optimization processes must be applied for restoring the mesh quality in the region of the cut. The local optimization is inexpensive, since it is applied to a surface mesh and to the layer of elements that share mesh entities with it.

Different implementations of the local procedures are possible. The following sequence of operations is robust and very effective in our experience. First, the list of newly generated mesh edges is scanned and the split location of the parent edge is examined. If the split location deviates from the edge mid point more than a preselected value, the edge is collapsed. Edge collapsing in three dimensions is exemplified in figure 1. The target vertex for the collapsing is always a vertex that belongs to the newly generated surface mesh, otherwise the surface mesh would be distorted. When both edge vertices belong to the surface mesh, the best target vertex for collapsing the edge is chosen based on the quality of the resulting retriangulations of the two possible polyhedra. Classification against the underlying solid model entities allows to ensure the topological validity of the collapsing [25].

To further improve on the quality of the mesh in the region of the cut, we apply a sequence of local optimizations in the following order: local edge swaps on the surface mesh, face and edge splits. The whole procedure is repeated until the local quality of the grid is restored.

We have now obtained a triangular grid that discretizes the surface approximating the feature that will be refined. The mesh entities of this grid are now duplicated and separated, generating a single layer of prisms. The required space for inserting the prisms is obtained by slightly deforming the rest of the mesh using a vertex repositioning algorithm based on a fictitious elastic problem, described in the following. The single layer is then split in order to create stacks of prisms, that are subsequently subdivided into tetrahedra. The whole procedure is exemplified in figure 2, that shows the generation of the surface discretization, its opening and the subsequent filling.

2.4. **Boundary Layer Meshing Algorithm.** For boundary layers, the procedure is normally driven by user-supplied input. Attributes specifying (possibly non-uniform) boundary layer thickness t, number of layers n_l and

their geometric distribution are associated with faces in the solid model wherever a boundary layer needs to be grown. The meshing process starts by duplicating the mesh entities classified on the model faces that have boundary layer meshing attributes. This creates two identical overlapping face discretizations: one, termed the "model face triangulation", will not be affected by the meshing algorithm and will provide the surface mesh on the model boundary; the other, termed the "transition triangulation", will be repositioned during the gridding process and will connect the initial isotropic grid to the newly formed boundary layer mesh.

At this step of the procedure, each couple of associated mesh faces defines a triangular prism of null thickness. Each prism is then subdivided into a stack of n_l prisms. Finally each prism in the stack is tetrahedronized, using the algorithm described in Section 2.4.1. The boundary layer mesh is then locally inflated, i.e. the transition triangulation is "pushed" away from the model face triangulation, with the goal of reaching the desired boundary layer thickness. The stacks of prisms follow the motion of the transition triangulation, providing a through-the-thickness discretization of the boundary layer.

The initial isotropic grid is deformed using a mesh repositioning scheme in order to accommodate the motion of the transition triangulation. The grid deforming algorithm uses fictitious springs that connect the mesh entities as described in Section 2.5. The scheme ensures that interpenetration of neighboring mesh regions will be avoided, and therefore the mesh validity will be guaranteed even for large motions. Furthermore, the scheme is based on force control, i.e. suitable forces are applied at the transition triangulation vertices so that the requested amount of displacement (layer thickness) is obtained at low-curvature points. The forces are applied along marching directions that are computed in order to guarantee the "visibility" of the repositioned vertex from all mesh faces in the vertex-manifold, as detailed in Section 2.6.1.

Note that only the vertices of the transition triangulation are forced to move along the prescribed marching directions. All other vertices on each growth curve, i.e. the lines that connect each vertex on the model face triangulation with its sibling on the transition triangulation, are free to be repositioned during the smoothing steps in the boundary layer mesh. This mesh relaxation, once again due to the presence of fictitious springs that connect entities in the grid, provides an automatic way of curving the growth directions in regions with closely spaced faces or of high curvature. Figure 3 shows the automatic curving of growth directions obtained in the common case of a corner region.

The procedure automatically deals with problem configurations, as for example concavities with high curvature, and avoids self-intersections when layers are grown on closely spaced model faces. In fact at problem locations, as for example points of high curvature or sharp corners, the applied force

will automatically generate smaller grid deformations than in "flat" locations, since the fictitious structure is locally stiffer. Applied forces are also adjusted by the algorithm according to the local characteristic dimensions of mesh regions, in order to reduce the final displacement in areas where elements are smaller, typically where curvature based refinement has been used by the isotropic mesh generator. Finally, in order to obtain a smoother variation of vertex displacements, applied loads are corrected through a distance weighted averaging among all vertices in a vertex-manifold. A typical grid resulting from this process is exemplified in figure 4.

In the case of the advancing layers method, self-intersections are usually created in regions of high curvature or at closely spaced faces. Fixing of boundary layer intersections must then be accomplished by shrinking and pruning of layers. Furthermore, a localization structure, e.g. an octree, is needed in order to identify all the self-intersecting elements avoiding $\mathcal{O}(n^2)$ operations. We note that the need for intersection checks and fixes is completely avoided with the present method, since the possibility of creating intersections is avoided a-priori.

The repositioning of the transition triangulation is usually performed in incremental steps, i.e. a first value of the load is imposed on the triangulation vertices and the isotropic and boundary layer grids are repositioned using a few smoothing steps of the mesh motion algorithm; next, a second load increment is applied on the vertices followed by smoothing. The process continues until the desired layer thickness is obtained. Throughout this process, checks are performed in order to ensure that the computed displacements do not cause a vertex to cross the planes identified by each mesh face of the polyhedral cavity containing it. If a plane crossing is identified, the applied load is decreased accordingly. Consequently, it is possible that a vertex does not reach the desired final displacement, stopped in its movement by a neighboring face. In order to avoid local abrupt changes in the displacements of the transition triangulation with consequent possible sever deformations of the mesh, which of course have a negative impact on the final result, it is beneficial to propagate these load changes to the applied loads of the neighboring vertices. This is here accomplished by a distance weighted averaging for all vertices in the vertex-manifold.

2.4.1. *Tetrahedronization.* Prisms in the anisotropic region are tetrahedronized. In fact, in general a hybrid mesh that consists only of tetrahedra in the isotropic part of the grid and of prisms in the anisotropic layer, cannot be guaranteed, unless suitable constraints on the definition of the layer growth attributes are specified, as detailed in Section 2.6.3. Clearly, the tetrahedronization has to be conforming, in the sense that prisms sharing a quadrilateral face need to use the same of the two possible splittings of that face. All possible tetrahedronization templates for a prism are given in figure 5. Each template can be symbolically represented by orienting the edges of one of the triangular faces of the prism, according to a conventional

rule, as shown in the figure. Note that no template induces a closed loop orientation of the edges.

This fact can be used for constructing a simple algorithm that ensures a correct tetrahedronization. To this end, the list of vertices in the model face triangulation is traversed. Each visited vertex orients all edges that use it and that have not yet been oriented. The orientation is always defined according to a pre-specified rule, e.g. in the direction pointing towards the vertex being examined. The procedure is exemplified in figure 6. Once all vertices in the surface mesh have been visited, all edges have been oriented. It is easily shown that no closed loop can be generated by such process. Therefore, the computed orientations can be used for determining the correct prism splitting template.

2.5. **Mesh Motion Algorithm.** The mesh motion algorithm is used for deflating the isotropic mesh. A suitable design of such algorithm is critical for the performance of the proposed method. The vertex repositioning problem must be solved within the following constraints. First, the grid must not be severely deformed in the proximity of the anisotropic layer in order not to degrade its quality. Second, it must provide an automatic way of dealing with problem regions, as mentioned earlier. Third, it must be robust and efficient, and in particular it must not generate invalid elements even for large amplitude motions. In order to satisfy the constraints, the following procedure was devised.

The algorithm is based on a fictitious elasticity problem, based on replacing the mesh edges with springs. Following ref. [3], the spring stiffnesses are inversely proportional to the edge lengths in the initial configuration, so that small elements tend to be very stiff while larger elements are softer. However, the method in its classical form can entangle the mesh when a vertex crosses a neighboring mesh face. This means that the classical algorithm can only be used for moderate mesh displacements. An improved method based on the use of additional torsional springs at the vertices was proposed in ref. [12] for two–dimensional grids. The torsional springs are designed so as to guarantee that neighboring triangles can not interpenetrate each other, in order to ensure that mesh entanglement will be avoided. However, the introduction of torsional springs renders the problem non-linear, so that a linearized version of the scheme must be developed. Furthermore, the method becomes quite cumbersome in three spatial dimensions.

Hence, we propose here a new method that is simple, yet guarantees valid meshes even for motions of large amplitude. The basic idea is to use the classical edge springs of ref. [3] together with additional linear springs that prevent a mesh vertex from crossing a neighboring face. For each region $\mathbf{r}$ using a vertex $\mathbf{v}$, a new linear spring is constructed that connects $\mathbf{v}$ with its projection $\mathbf{v}'$ on the plane of the face $\mathbf{f}$ of $\mathbf{r}$ opposite $\mathbf{v}$. This additional spring is exemplified in figure 7 for a single tetrahedron connected to a vertex, for clarity.

Once the additional springs are constructed for all tetrahedra using a vertex, one has effectively constrained the same vertex not to leave the polyhedral ball that encloses it, therefore ensuring a final valid mesh. Furthermore, the presence of the additional springs is also beneficial in terms of mesh quality, since they will tend to keep each vertex closer to the centroid of the ball, pushing it away from its boundary. The resulting mesh repositioning scheme is linear, and can be straightforwardly modified to handle the two–dimensional case.

In classical mesh motion problems, the displacements at an interface, e.g. a fluid/solid boundary, are imposed at the grid vertices at that interface. The displacements are then propagated to the rest of the grid using some form of smoothing. This ensures that the required displacement is always achieved, at least as long as the smoothing algorithm does not entangle the mesh. Such an approach is unsuitable for the present application. A better way of deforming the mesh is through an algorithm based on force control, not displacement control. In practice, a force is applied at mesh vertices on the transition triangulation along a suitable local direction. This force is calibrated in such a way as to produce the required local displacement in a nearly flat boundary case. When the boundary presents sharp corners or high curvature, the local forces are not able to deform the mesh to the extent of producing the required displacement, since high local deformations would be required. These deformations are resisted by the stretching and compressing of the linear springs.

Care is exercised in order to ensure the visibility of the displaced vertices from their parents, and for guaranteeing that the deflation process advances along directions close to the local normal. The computation of local growth directions is discussed in Section 2.6.1.

The smoothing process is based on a vertex-by-vertex relaxed Gauss-Seidel scheme. Vertices in the isotropic region are ordered using the algorithm described in Section 2.5.1, with the goal of maximizing for each newly processed vertex the number of already processed adjacent vertices. This helps in propagating effectively the displacements throughout the mesh even with a small number of iterations.

The same smoothing is applied in the isotropic and anisotropic parts of the grid. However, while in the isotropic grid the goal of smoothing is that of deforming the mesh in order to accommodate the mesh layer, in the anisotropic mesh it determines the distribution law of layer thicknesses normal to the wall. In fact the transition triangulation, pushed by the applied loads, pulls with it the various layers of prisms away from the model boundary, layers that, at the beginning of the process, are of null thickness and are all collapsed onto the model face triangulation. This pull is realized by the smoothing process in the anisotropic grid. A small number of iterations will produce thinner layers close to the model face, approximating a geometric distribution, while a larger number of iterations will tend to create an equally distributed spacing between layers. This is easily seen

in a one-dimensional case, where one has a string of points connected by equal springs. The points are initially all coincident. Next, the first point is displaced by a distance d. All other points are then processed using a Gauss-Seidel iteration. The second point in the string is repositioned, and its equilibrium position is at $d/2$. Next, the third point is considered, and its equilibrium position is found at $d/4$, and so on. When one full Gauss-Seidel iteration has been performed, all points have been processed once and their positions are given by a geometric distribution. Increasing the number of Gauss-Seidel iterations will tend to equidistribute the point positions, yielding equal spaces between them. It is clear that other distributions could be obtained using progressively stiffer or softer, rather than constant, springs connecting the points. This simple one-dimensional model approximates the behavior of the grid vertices along a growth curve in the anisotropic layer mesh. Note that in this region, different numbers of smoothing steps can be used along growth curves or in orthogonal directions. In fact, while the former controls the layer spacings, the latter curves the growth directions and might require a different number of smoothing passes to yield a satisfactory solution.

Using the advancing layers method, one explicitly constructs the layers according to a specific spacing function and would therefore probably not use a smoother inside the boundary layer region. This approach on the one hand gives total control on the layer thicknesses, but at the same time makes a little harder to guarantee the element quality since there are few degrees of freedom to play with, especially if one does not use curved growth directions. On the contrary, it is clear from the previous discussion that the proposed method can only approximate the spacing function using the smoother. The control on the layer thicknesses is then somewhat reduced in the present case when compared with the advancing layers technique. However, it should be remarked at this point of the discussion that one characteristic of the proposed approach is that all the basic steps of the generation of an anisotropic layer region, namely computation of the growth directions and of the growth displacements, curving of growth curves, computation of the grid spacing normal to the wall, etc., are now all blended in one single process, the mesh smoothing. This means that the various steps are not realized in any particular order, as with other algorithms, but are all coupled together and in this sense each one of them affects all the others. Therefore, while one is pushing on the transition triangulation for making room for the anisotropic layer, one is also relaxing the positions of the new grid points within the layer according to the fictitious elasticity problem solved by the smoothing algorithm, achieving at once all the various necessary goals. While on the one hand this necessarily reduces somewhat the available control as previously noted, the intimate coupling of the basic logical steps of the mesh generation process implied by this approach would seem to offer the potential for a beneficial effect on the quality of the resulting grids.

2.5.1. *Mesh Ordering Algorithm.* For efficiently propagating the information during the smoothing process, vertices need to be ordered. The problem is somewhat similar to the generation of the line orderings used by relaxation algorithms for the iterative solution of discrete operators on unstructured grids. In view of the fact that a very limited number of smoothing steps will be performed for efficiency reasons, we would desire to maximize for each vertex the number of already processed neighbors.

A simple ordering scheme was devised for this application, based on a modified greedy algorithm [13]. First, model faces with growth attributes are collected in connected sets, i.e. if two model faces share a common model edge or vertex they are assigned to the same set. Next, a mesh vertex classified on model boundary is selected for each connected set. This vertex represents the "seed" point for an ordering process. The seed vertex is assigned the number 1 and labeled $\mathbf{v}_1$. Starting from $\mathbf{v}_1$, all edge-connected vertices that are also classified on the connected set or its boundaries are gathered, numbered and stored in a linked list. The vertex that has been assigned the number 2, $\mathbf{v}_2$, is now considered. The process is repeated, and all edge-connected vertices to $\mathbf{v}_2$ that are also classified on the connected set or its boundaries are collected in the list and numbered. At the end of this first phase, all the numbered vertices are on the transition triangulation, and therefore are all directly affected by the imposition of the loads that drive the smoothing procedure. The process is now restarted from vertex $\mathbf{v}_1$. All edge-connected vertices that have not yet been numbered are gathered, numbered and stored in the linked list. Next, vertex $\mathbf{v}_2$ is considered, and all its edge-connected vertices that have not yet been numbered are processed. The procedure continues until all vertices have been numbered.

The algorithm essentially constructs successive layers of vertices, the first layer being the transition triangulation, that wrap the connected set and march away from it. The procedure is illustrated in figure 8. A typical example of the ordering constructed in this fashion is given in figure 9 for the two-dimensional mesh of a submarine model. The growth set is composed of all the model edges that form the submarine silhouette, while the seed point is labeled P. Colors represent the numbers associated with the vertices by the numbering process. Note how the ordering creates layers that enclose the model boundary propagating outwards towards the rest of the domain. The described process is straightforwardly generalized to handle the internal layer case.

2.5.2. *Improved Centroidal Smoothing.* An additional way of locally improving the grid quality is provided by some variant of the well known Laplacian smoothing process [14]. A general form of the scheme can be written as

$$P' = \frac{\sum_i^n w_i P_i}{\sum_i^n w_i}, \tag{1}$$

where P is the point associated with a given vertex $\mathbf{v}$ and the P_i's are the points associated with the vertices that are edge-connected to $\mathbf{v}$. This form of local smoothing was used in the present work as a further help in trying to remove problematic configurations.

This method is quite effective for improving the grid locally. However, some care should be exercised since the polyhedral region defined by the P_i's might be non-convex. This situation is most likely to happen in highly distorted regions, or in the proximity of closely spaced faces and corners, exactly where the need for local improvement is the highest. Since even a non optimal repositioning might lead to some local improvement in these pathological situations, a modified strategy was devised. The basic idea is to "regularize" the problem by constructing a convex region around vertex $\mathbf{v}$ using some new points $\tilde{P}_i$ instead of the vertex points P_i. At this point, the update defined by (1) can be used safely.

To present the main idea behind the proposed algorithm, we can start from the simpler two-dimensional case. The edge-connected points P_i define in this case a polygon. Each polygon edge divides the plane into two semi-planes, one that contains $\mathbf{v}$ and that therefore defines valid repositionings of the same vertex, and a second one that will produce invalid repositionings. The "right" final convex polygon for the safe application of one pass of Laplacian smoothing is now given by the boundary of the figure obtained by intersecting all the valid semi-planes. This is exemplified in figure 10 for a simple concave configuration. In reality we do not need to define the real edges of this new regularized polygon, but only its vertices, since only these are needed for doing Laplacian smoothing.

The position of these vertices can be practically computed by considering subsequent intersections between couples of edges of the original polygon. For a convex polygon, all these intersections coincide with the vertices of the polygon itself. However, for a non-convex figure, additional points will be generated. All points need now to be checked against all polygon edges to determine whether they are on the valid semi-plane or not. This is easily determined by defining an arbitrary normal to the edge and considering two distance vectors, one between one of the edge vertices and the vertex that will be repositioned, $\mathbf{v}$, and the second between the same edge vertex and the point that needs to be checked. One then computes the dot products of the edge normal with each distance vector; if the two products have the same sign, the checked point and vertex $\mathbf{v}$ are on the same semi-plane and the point will be retained, otherwise they are on different sides and the point needs to be rejected. The process is continued untill all points have been checked. At the end of the procedure, the remaining points define the regularized polygon.

In the three-dimensional case, the same idea can be generalized by considering now all points generated by the intersections of all the edges of the surface of the polyhedral region with all its triangular faces. Once again,

each newly generated point needs to be checked to see whether it lies on the valid semi-space with respect to vertex $\mathbf{v}$.

2.6. Implementational Issues.

2.6.1. *Computing Deflation Directions.* Marching directions are based on the "normal" to the vertex-manifold, i.e. to the group of mesh faces using the vertex and classified on model faces or on the cutting surface. In fact, in order not to create invalid elements, the new vertex position must be visible from all faces in the manifold. We note that we could use the normals to the model faces rather than the mesh faces when growing boundary layers, since the procedures have access to the true problem geometry through a solid modeling system. However, for coarse meshes this might violate the visibility condition. In this work, we use the method suggested in ref. [18] for computing the initial values of the directions. The algorithm first renders the problem two-dimensional by finding the bisection plane for the wedge identified by the two faces $\mathbf{f}_1$ and $\mathbf{f}_2$ in the $\mathbf{v}$ vertex-manifold that form the smallest dihedral angle. Then, the visibility angle on that plane is bisected to identify the manifold normal. A visibility cone can now be constructed at $\mathbf{v}$, tangent to $\mathbf{f}_1$ and $\mathbf{f}_2$.

In regions with high curvature, at corner points or in proximity of model faces forming small dihedral angles, it is beneficial to let the marching directions slightly deviate from the manifold normals computed as explained above. This reduces the convergence or divergence of the growth curves, therefore minimizing the local compressing and stretching of the transition triangulation. To this end, the nominal normals are smoothed by distance-weighted averaging. Given a vertex and its normal, first all normals at the vertices in the vertex-manifold are smoothed, i.e. a first ring of normals around the vertex is processed. Next, all faces adjacent to the vertex-manifold are considered, and their normals at vertices are smoothed, i.e. a second ring of normals around the vertex is processed. The process is continued until a preselected number of rings (typically four or five) has been affected by the smoothing. After smoothing at each vertex, the new marching direction is verified to lie within the visibility cone, in order to ensure the validity of the subsequent repositioning.

We note once again that the deflation directions determine the marching vectors of the sole vertices in the transition triangulation. All other vertices within the anisotropic layer, i.e. all vertices along the growth curves, will be allowed to deviate from these directions according to the layer smoothing process. This will provide an automatic way of curving the growth curves according to the local problem geometry.

2.6.2. *Mesh Quality Enhancements.* Mesh smoothing is in some sense limited by the topology of the grid, since it only deals with the repositioning of the vertices, and can not change the local connectivity. In order to relax this constraint and to further ensure that the isotropic grid quality is not

affected by the repositioning process, local retriangulation operations are recursively applied where needed. The local optimization criterion is based on the largest dihedral angles of each tetrahedron. The retriangulation does not affect the topology of the transition triangulation, not to interfere with the highly stretched elements in the boundary layer, unless severe local deformations can not be corrected through the already described repositioning procedures.

Five basic retriangulations are considered: edge removal and its inverse operation multi-face removal, edge swapping, edge collapsing, and entity (region, face and edge) splittings [11]. As a first step, the dihedral angles of all elements considered for optimization are calculated. Each element violating a user–defined threshold value is put into a linear list. Depending on the configuration of each element in the list (number of dihedral angles above the threshold value, topological or geometrical constraints, etc.), a suitable subset of the above mentioned optimization procedures is applied to eliminate that element in favor of improved elements. The procedures might fail for a specific element if the resulting configuration is topologically or geometrically not valid, or if they lead to a degradation of the quality of any element involved in that local retriangulation. In this case, or if the element is improved but the largest dihedral angle is still above the threshold value, the element is considered for improvement in a second pass, after all elements have been processed. Since the neighborhood of the elements that failed in the first pass may have been modified, it is possible that they are fixed in a second pass. The procedure is repeated until a given threshold value is reached or no further improvement can be achieved.

Local coarsening can also be used in a different context. In fact, if boundary layers need to be grown on closely spaced model faces facing each other, the squeezing and compressing of the isotropic grid could not allow to reach the desired layer thicknesses. In this case, local coarsening can be applied in the isotropic region, for easing the growth process.

2.6.3. *Ending Layers.* In the case of hybrid/prismatic mesh generators, layers are grown around all boundary faces of a body. In many applications however, boundary layers are needed only on a limited set of the model faces. The procedure here described can deal with situations involving adjacent faces with different boundary layer growth attributes. Figure 11 shows the common case of a boundary layer whose thickness goes to zero at a model edge. This is simply handled by specifying a null growth for the mesh vertices classified on the model edge, followed by the already described prism splitting and tetrahedronization. A subsequent pass of mesh collapsing eliminates all edges of zero length, providing the required grid. Figure 12 shows the other common case of a boundary layer ending at neighboring model faces. Interaction with the solid modeler ensures the correct positioning of the boundary layer vertices on curved model faces, as detailed in Section 2.6.5. The selection between an edge-ending layer (figure 11) or a

face-ending layer (figure 12) is determined based on the maximum allowable dihedral angle in the mesh, typically 160 deg.

Clearly, if all model faces are tagged with a boundary layer, or if all grown layers end on model faces as in the case of figure 12, the final prism tetrahedronization can be skipped, and one obtains a mixed hybrid/prismatic grid.

There are however situations that require special attention. Figure 14(a) gives an example of a possible problematic configuration: in this case a boundary layer will be grown on face $\mathbf{F}_1$, but not on faces $\mathbf{F}_2$, $\mathbf{F}_3$ and $\mathbf{F}_4$. One possible solution would be to allow for two growth curves at vertex $\mathbf{V}_1$, one classified on edge $\mathbf{E}_1$ and one on edge $\mathbf{E}_2$. A simpler solution is sketched in figure 14(b) and (c). In essence, the boundary layer is allowed to spillover to the neighboring faces in a transition region of thickness comparable to that of the boundary layer itself. Figure 14(b) shows the patch of mesh faces involved in the process. Figure 14(c) shows the detachment of the patch of faces from the model boundary, and the creation of the boundary layer region. Now all boundary layer elements belong to one of the two cases of figure 11 and 12, so that the standard procedures can be used.

Another example of anisotropic layer boundaries that need special handling is represented by the case of an internal layer that ends within the domain, i.e. it does not extend all the way to a model boundary. A common such case is encountered, for example, when refining a shock wave at the tip of a blade; the anisotropic layer in this case will end at a certain distance from the blade surface, close to the shock boundaries, and will clearly not have to extend to the far field for efficiency. In this case, the layer thickness will have to be null at its internal boundary, similarly to the previously discussed boundary layer case with zero thickness at a model edge. Here again a null growth length is specified for the mesh vertices at the internal edge of the layer, followed by the already described prism splitting and tetrahedronization. A subsequent pass of mesh collapsing eliminates all edges of zero length, providing the required grid. The result of such procedure is exemplified in figure 13.

2.6.4. *Reclassification.* Certain applications will require the entities of the final adapted grid to be completely classified, for example when the analysis attributes (boundary conditions, loads, material properties, etc.) are associated with entities in the solid model, so that they can be consistently transferred to the new adapted grid for use by the solver. Furthermore, a correct classification information is necessary for repositioning the mesh vertices on curved model boundaries in order to provide geometrically consistent grids and avoid loss of precision, as detailed in Section 2.6.5. Because of these reasons, it is necessary to compute the classification of all newly created mesh entities.

Vertices classified on a model entity of degree k are always reclassified on an entity of degree $k+1$. Therefore, vertices classified on model faces are

always classified internal. For vertices classified on model edges, one has to check whether both model faces that use that edge are associated with a boundary layer attribute. In case they both have attributes, the vertex is classified internal, otherwise it is classified on the model face that does not have the boundary layer attribute. This situation is exemplified in figure 15. Mesh $\mathbf{v}_1$ is classified on model edge $\mathbf{E}_1$, which is used by face $\mathbf{F}_1$ and face $\mathbf{F}_3$. Since a boundary layer is grown only on $\mathbf{F}_3$ and assuming a face-ending layer, the newly generated vertex $\bar{\mathbf{v}}_1$ is necessarily classified on $\mathbf{F}_1$. Similarly, for vertices classified on model vertices one has to find the model edge that is used by model faces that are not tagged for boundary layer growing. Again with reference to figure 15, the newly generated vertex $\bar{\mathbf{v}}_2$ is necessarily classified on model edge $\mathbf{E}_2$.

2.6.5. *Dealing with Curved Boundaries.* For accuracy, it is critical that newly generated or repositioned mesh vertices are placed on the true model boundary. The classification information together with the direct interfacing of the procedures with a solid modeler, provide for a straightforward solution to this problem. In fact, all mesh entities, including the ones generated in the boundary layer, are classified against the various model entities, as discussed in Section 2.6.4. Then, after each smoothing step, each mesh vertex classified on model face or model edge is snapped to the boundary. The new location is provided by the solid modeler through a closest point interrogation. The position of the mesh vertex before the smoothing pass is used as the initial guess for the interrogation.

It should however be noted that the simple snapping of vertices to the boundary can in some cases generate invalid or severely distorted elements. More sophisticated techniques can be used for dealing with this issue, for example by applying local retriangulations as suggested in ref. [20].

2.7. **Numerical Examples.** In this section, two test cases demonstrate the developed procedures for the refinement of boundary features, while a third case shows the insertion of internal anisotropic layers. More examples and additional details can be found in refs. [6, 7].

2.7.1. *Mesh Quality Measures.* For the examples here considered, mesh quality measures are given in terms of geometric criteria. This allows to determine the grid characteristics in a solver and application-independent manner, while actual numerical simulations would raise questions concerning the particular finite element or finite volume formulation, boundary conditions used and other problem dependent issues.

Two quality measures are considered in the following. The first is the maximum dihedral angle between two neighboring mesh faces, $Q_\delta = \max_{i=1,6} \delta_i$, where the angle at edge i shared by faces j and k is $\delta_i = \pi \pm \arccos(\vec{n}_j \cdot \vec{n}_k)$, with $\vec{n}_l$ the normal to face l. Large dihedral angles can negatively impact the solution of partial differential equations, affecting the discretization error and the conditioning of the discrete problem [2]. The second quality measure

is the ratio of the radii of the inscribed and circumscribed spheres, $Q_s = r/R$, and gives a measure of the stretching of an element. The boundary layer meshing process should not significantly affect both measures for the incoming isotropic mesh, while it should generate elements in the anisotropic regions that have the largest dihedral angle close to $\pi/2$, and that match the requested value of stretching.

2.7.2. *Nacelle.* The first example deals with the growth of a boundary layer on an aeronautical engine nacelle. For this problem 10 anisotropic layers were grown on the nacelle model faces, with geometric thickness distribution in the direction normal to the wall. A constant boundary layer thickness was requested, implying a value of stretching ranging from about 5 to about 40. Stretching varies because of non-uniform layer thickness and because the model face triangulation is graded due to curvature based refinement. The initial isotropic grid had 119240 tetrahedra, and 5318 mesh faces classified on model faces with growth attributes. The final resulting grid has 429020 tetrahedra, and is depicted in figure 16. Elements in certain parts of the boundary layer were removed to expose the anisotropic grid and the underlying model face triangulation. Elements in the isotropic part of the mesh are not shown for clarity.

Figure 17 shows a histogram of the maximum dihedral angles for the initial and adapted grids. For the adapted grid, only dihedral angles for tetrahedra in the isotropic region are shown. Therefore, the figure compares the quality of the isotropic grid as produced by the mesh generator and after it has been affected by the mesh repositioning required for inserting the boundary layer. It appears from the plot that the distribution of angles is even better for the adapted than for the initial grid.

Figure 18 shows a histogram of the maximum dihedral angles for the sole anisotropic part of the adapted grid. Note how the maximum angles are clustered around $\pi/2$ in this region, deviations from this value being due to the regions of high curvature close to the lip of the nacelle. Figure 19 shows a histogram of the ratio of the radii of the inscribed and circumscribed spheres, Q_s, for the sole anisotropic part of the adapted grid. The stretching measures are clustered around the range of expected values, from 5 to 40.

2.7.3. *Submarine.* The second example shows the growth of a boundary layer around a more topologically complex submarine model. In this case 6 anisotropic layers were grown on the submarine model faces, with geometric thickness distribution in the direction normal to the wall. The initial isotropic grid had 165305 tetrahedra, with 6994 mesh faces classified on model faces with growth attributes. The final grid has 291197 tetrahedra, and is shown in figure 20. Even in this case, certain parts of the boundary layer were removed to expose the anisotropic grid and the underlying model face triangulation.

Figure 21 shows details of the anisotropic grid. Both the junction of the leading edge of the fin with the submarine tower and the tip of the fin

are shown. Various features of the proposed procedures are simultaneously visible, in particular the tilting of growth directions, the curving of growth curves and the variable boundary layer thickness.

Even in this case, the quality of the initial grid is not significantly affected by the repositioning required for the insertion of the boundary layer. Figure 22 shows a histogram of the maximum dihedral angles for the initial and adapted grids, in this latter case considering tetrahedra in the isotropic region only.

2.7.4. *Insertion of a Thin Layer.* The third example concerns the introduction of an internal anisotropic layer. We consider a simple cubic domain with a quarter sphere cut-out at one of the corners. First, a boundary layer mesh is grown on the sphere surface, as previously explained. Next, we insert a thin layer of elements in front of the sphere using the internal layer algorithm detailed in the previous pages. This example is entirely of an academic nature, especially since no particular attention was paid to produce the necessary mesh gradation for a realistic computation, but might be representative of the refinement of a strong shock in front of a blunt body.

Figure 23 shows a view of the internal surface of the sphere, of the anisotropic layer and of the isotropic mesh. Elements in the isotropic and anisotropic grids were removed in order to show the mesh interior. Figure 24 offers a slightly different view of the same problem, where now only elements in the isotropic grid were removed.

Although the problem topology is quite simple in this case, this problem shows some of the typical difficulties encountered with more realistic applications, namely the arbitrary cut of elements and the collision of the cut front with the model boundaries. Even in this case, it was observed that the mesh quality measures remained essentially unchanged by the proposed procedure in the isotropic mesh, while the anisotropic grid is of very high quality with virtually no large dihedral angles.

3. Procedures for Anisotropic Error Estimation

3.1. Introduction and Motivation. The a-posteriori estimation of the error associated with finite element solutions is still an area of very active research. Among the many procedures proposed, we can mention here residual-based methods and various types of recovery-based estimators (see refs. [27, 1] for a review of available techniques).

A procedure that is simple and yet very effective is the Zienkeiwicz-Zhu (ZZ) error estimator [29, 30], that reconstructs smoother gradients than the ones associated with the numerical solution of the finite element problem. Various possible implementations of the method used to recover gradients are possible and have been proposed and tested in the past, for example the superconvergent patch recovery (SPR), the Clèmant interpolation, and various form of averaging [19, 24].

Usually these procedures amount to averaging the piecewice constant gradient ∇u^h over suitable patches of elements around a given node (patch assembly point), in order to obtain a continuous piecewise linear recovered gradient. This recovered value is sometimes found to superconverge to the gradient of u on a set of points, which coincide with the patch assembly point when using linear elements.

The concept of effectivity index is frequently used to measure the quality of an error estimator. The effectivity index is defined as

$$\theta = \frac{\eta}{\| e^h \|},$$

where η is the global error indicator

$$\eta = \left(\sum_{K \in \mathcal{T}_h} \eta_k^2 \right)^{1/2},$$

over the triangulation $\mathcal{T}_h$ of the domain Ω, η_K being the elemental error indicator. An error estimator is said to be asymptotically exact if $\theta \to 1$ when the mesh size h approaches zero. There is a relationship between the accuracy of the recovered gradients and the effectivity index, and in fact if the gradient recovery is superconvergent, the error estimator is asymptotically exact [29, 30]. The ZZ estimator is indeed asymptotically exact, if the solution is sufficiently smooth and the grid possesses certain properties of regularity [24]. In practice, the ZZ estimator has been applied to more general unstructured grids with interesting results [30]. This indeed motivates the use of the ZZ procedures even in the context of the applications that are releveant to this research project.

It should be mentioned that in reality, for realistic adaptive applications where the grid size will clearly not tend to zero, or on highly anisotropic grids, the requirement of asymptotical exactness is more appropriately replaced by a saturation assumption [27], which can be expressed as

$$\| \nabla^{ZZ} u^h - \nabla u \|_{L^2(\Omega)} \leq \beta \| \nabla u^h - \nabla u \|_{L^2(\Omega)},$$

with $\beta \in [0,1)$, where $\nabla^{ZZ} u^h$ is the ZZ recovered (improved) gradient.

In this work we use the ZZ estimator for driving the adaption process. At first, we have generalized the ZZ method to the compressible Euler and Navier-Stokes equations written in terms of entropy variables. This yields a numerical procedure that is directly usable per-se and that has been implemented in our parallel adaptive software based on the Galerkin Least-Squares finite element formulation [9, 8]. This formulation of the estimator is briefly described in Section 3.2. Next, we try to go beyond the simple use of the ZZ estimator, and we present some preliminary work on the coupling of the ZZ recovery with an estimate in the energy norm of the interpolation error which takes into account explicitely the anisotropic information of the solution and of the mesh elements. This part of the work is described is Section 3.3.

3.2. **An Implementation of the ZZ Estimator for the Compressible Euler and Navier-Stokes Equations.** The Navier-Stokes equations can be written as

$$\widetilde{\mathbf{A}}_0 \mathbf{V}_{,t} + \widetilde{\mathbf{A}}_i \mathbf{V}_{,i} = \left(\widetilde{\mathbf{K}}_{ij} \mathbf{V}_{,j}\right)_{,i} + \mathbf{F}. \tag{2}$$

Denoting by $\mathbf{U}$ the conservative variables, $\mathbf{V} = \partial H / \partial \mathbf{U}$ are the entropy variables, defined starting form a generalized entropy function $H = H(\mathbf{U})$. Furthermore, $\widetilde{\mathbf{A}}_0 = \partial \mathbf{U} / \partial \mathbf{V}$ denotes the associated Reimannian metric tensor, which can be used for defining consistent energy norms.

To discretize the Navier-Stokes equations in space and time, we use the Time-Discontinuous Galerkin finite element method, stabilized using the least-squares formulation of ref. [26]. The spatial discretization uses linear shape functions.

According to the ZZ method, the energy norm of the error is then computed as

$$\| \mathbf{e}^h \| = \left(\int_\Omega \left(\nabla^{ZZ} \mathbf{V}^h - \nabla \mathbf{V}^h \right) \cdot \mathrm{diag}[\widetilde{\mathbf{A}}_0] \left(\nabla^{ZZ} \mathbf{V}^h - \nabla \mathbf{V}^h \right) \mathrm{d}\Omega \right)^{1/2}, \tag{3}$$

where

$$\mathrm{diag}[\widetilde{\mathbf{A}}_0] = \begin{bmatrix} \widetilde{\mathbf{A}}_0 & & \\ & \cdots & \\ & & \widetilde{\mathbf{A}}_0 \end{bmatrix}$$

and

$$\nabla(\cdot) = \begin{bmatrix} \dfrac{\partial(\cdot)}{\partial x_1} \\ \vdots \\ \dfrac{\partial(\cdot)}{\partial x_d} \end{bmatrix},$$

where d is the number of space dimensions. $\nabla^{ZZ} \mathbf{V}^h$ is the recovered gradient of the entropy variables, which is here computed based on averaging using element volumes. This technique is here favoured to the alternative offered

by least-squares fitting. In fact, it is not uncommon in realistic applications to incur in local mesh configurations that will make the least-squares reconstruction singular or very ill-conditioned. For example, this will happen every time one has the centroids of the patch of regions that lie on the same plane.

The implementation of the formulation is straightforward and was easily parallelized using the message passing protocol MPI, as for all the rest of the code.

This form of the estimator can be used for driving isotropic adaption processes. Elements where the local percentage error is above a user-determined threshold are refined. In this work we use edge-based mesh modification primitives, and therefore the elemental errors are reinterpolated onto the mesh edges before adaption. In turn, edges marked for refinement are split using all possible tetrahedral edge-splitting templates. Alternatively, edges can be collapsed if local coarsening is requested.

3.2.1. *Numerical Examples.* We report here two simple numerical experiments that were used, among others, to assess the correct implementation of the methodology.

The first example is a two-dimensional steady problem consisting of a Mach two flow over a wedge at an angle of 10 deg, resulting in the creation of an oblique shock at the wedge leading edge. Since the code only supports three-dimensional elements, the problem was solved adding a third dimension equal to one tenth of the domain size. Figure 25 gives the true (at left) and computed (at right) error on the initial coarse grid. Good matching between the two quantities can be observed at all locations in the regions of the shock. Next, an automated adaptive analysis was started, using the computed ZZ error to drive the mesh refinement as previously explained. Figure 26 shows the obtained grid after two refinement steps. A nice and well distributed clustering of elements can be observed in the region of the shock.

Next, we consider the classical Onera M6 wing in transonic fligh. The wing is characterized by an aspect ratio of 3.8, a leading edge sweep angle of 30 deg, and a taper ratio of 0.56. The airfoil section is an Onera D symmetric section with 10% maximum thickness-to-cord ratio. We consider a steady flow problem characterized by an angle of attack $\alpha = 3.06$ deg and a value of $M = 0.8395$ for the freestream Mach number. In such conditions, the flow pattern around the wing is characterized by a complicated double-lambda shock on the upper surface of the wing with two triple points.

Figure 27 shows the error computed by the implemented error estimator on the initial grid. The error correctly identifies the region of the shock, and also targets the leading and trailing edges for refinement. Finally, figure 28 shows the adapted mesh obtained after two analysis-estimation-adaption cycles.

3.3. An Anisotropic Recovery-Based A-Posteriori Error Estimator. In this section we report an alternative use of the ZZ recovery technique, that aims at developing a true anisotropic estimator. It should in fact be noticed that the ZZ method, in its original formulation, does not directly allow for the construction of a local metric [16] that can then be used for driving an anisotropic adaption procedure. The basic idea of the formulation is briefly described in the following, and more details can be found in ref. [22] and the literature cited therein.

At first, we define a linear mapping that associates the reference element $\widehat{K}$ to the actual element K:

$$\vec{x} = M_K \hat{\vec{x}} + \vec{m}. \tag{4}$$

The non-singular Jacobian M_K can be factored by a polar decomposition as

$$M_K = B_K R_K, \tag{5}$$

where B_K is symmetric and positive definite, while R_K is orthogonal and amounts to a rotation. The eigenvalues of B_K are denoted $\lambda_{i,K}$, $i = 1, d$ and their associated eigenvectors are denoted $\vec{a}_{i,K}$, $i = 1, d$.

It is possible to prove under appropriate conditions that, given the bilinear form $B(u^h, v^h)$ corresponding to the weak form of, e.g., a diffusion problem, the bilinear form $B(e^h, v^h)$, where $e^h = u - u^h$ is again the discretization error, can be bounded as follows

$$\|B(e^h, v)\| \leq \sum_{K \in \mathcal{T}_h} \alpha_K \rho_K(u^h) w_K(v). \tag{6}$$

The first quantity appearing in the previous expressions can be synthetically given as

$$\alpha_K = \alpha_K(\lambda_{i,K}),$$

while the second term is

$$\rho_K(u^h) = \rho_K(r_K(u^h), R_K(u^h), \min_i(\lambda_{i,K})).$$

In the previous expression, $r_K(u^h)$ denotes the elemental internal residuals, while $R_K(u^h)$ represents the elemental boundary residuals. Finally, the last term writes

$$w_K(v) = w_K(G_K(v), \lambda_{i,K}, \vec{a}_{i,K}),$$

where $G_K(v)$ is a symmetric non-negative matrix defined as

$$G_K(v) = \sum_{T \in \mathcal{P}_h} \int_T \nabla v \otimes \nabla v \, \mathrm{d}T,$$

where $\mathcal{P}_h$ is the patch of all elements that share a vertex with element K.

These results can be used for computing a-posteriori error estimates based on the solution of the dual problem. This can be very expensive for non self-adjoint problems. A more interesting idea is to set $v = e^h$ in (6), to get an

implicit estimate for the square of the energy norm of the discretization error as

$$\| e^h \| \leq \sum_{K \in \mathcal{T}_h} \alpha_K \rho_K(u^h) w_K(e^h). \tag{7}$$

Since the right hand side of (7) still depends on the error, we need to replace $w_K(e^h)$ with a computable quantity. An elegant solution is to use the ZZ recovery technique to replace $G_K(e^h)$ with a new quantity $G_K(e^{ZZ,h})$ defined as

$$G_K(e^{ZZ,h}) = \sum_{T \in \mathcal{P}_h} \int_T (\nabla^{ZZ} v^h - \nabla v^h) \otimes (\nabla^{ZZ} v^h - \nabla v^h) \, \mathrm{d}T.$$

Finally we get a computable a-posteriori recovery-based error estimator defined on each mesh element as

$$\eta_K^2 = \sum_{K \in \mathcal{T}_h} \alpha_K \rho_K(u^h) w_K(e^{ZZ,h}). \tag{8}$$

This estimator provides anisotropic information in a consistent manner, since it is based on the ZZ recovery that is known to perform very well in a number of circumstances, and is also cheap to compute.

Ref. [22] gives details on how it is possible to reconstruct a local metric based on the estimator, in order to drive an anisotropic mesh modification procedure. In particular, a piecewise constant metric matrix can be computed on each element of the triangulation based on the knowledge of its eigenvalues and eigenvectors. These quantities can be computed based on the estimator (8), provided certain requirements on the error distribution and on the grid are specified. In particular, it makes sense to request that the error be equidistributed throughout the grid and that the number of elements is minimized. This leads to a minimization problem that gives the required definition of the metric matrices.

Preliminary results concerning the proposed technique are presented in ref. [22] on academic test cases.

4. Conclusions

In this document, we have summarized the research conducted to date on anisotropic adaption using unstructured finite elements, under the support of the European Research Office of the U.S. Army. This work was focused in two main areas: the development of anisotropic mesh adaption procedures and the problem of a-posteriori error estimation. These two points are also reflected in the two tasks (1.1 and 1.2) of the present contract, which read:

- **Task 1.1:** Analysis and development of anisotropic error estimation procedures.
- **Task 1.2:** Analysis and development of anisotropic mesh refinement strategies.

The delivarable for this part of the work stated:

> **Delivarable** (12 month milestone): technical document reporting on the results of the research in the field of anisotropic error estimation and mesh adaption, suitable for rotorcraft problems. The document will also detail the results gathered during preliminary testing and implementation of the methods, with the goal of determining their suitability for the application to full scale hovering rotorcraft problems.

Regarding the first area of investigation, we have here presented a procedure for the insertion of anisotropic tetrahedral grid layers in the interior and in the proximity of the boundary of curved three-dimensional domains. The present approach has some strenghts and some weaknesses, as probably most methods do.

Among the strong points, we can mention here that it allows to generate locally adapted anisotropic meshes with excellent control of the grid quality. In particular, since the anisotropic layers are "almost" structured in nature, the stretched elements are denoted by dihedral angles which are either very small or very close to ninety degrees. Theoretical results show that large dihedral angles affect the discretization error and the conditioning of the discrete problem, and should therefore be avoided. It should be noted that other anisotropic refinement procedures, such as some form of point insertion based on local anisotropic metrics [5], can not guarantee the control of large dihedral angles and in fact in general will produce very poor grids with respect to this quality measure.

The present method allows to use any existing mesh generator for producing the isotropic grid, and is not limited to advancing front methods as in the advancing layers procedure. Furthermore, the method does not require a complex post-processing to check and fix cross-overs and self-intersections. The software procedures are integrated with a solid modeler, that provides the required geometric and topological information.

On the other hand, it is also clear that, by its very design, the present approach is only suited to the refinement of one-dimensional anisotropic

features, i.e. for solution features that are characterized by a single local direction of high gradients, as it is the case of boundary layers and shocks. The generalization of the ideas here reported to also support the refinement of two-dimensional features, such as vortices, leads to very complicated meshing processes which are probably difficult to justify for practical applications. In this sense, it seems that a better approach would be to complement the present procedures with a metric-based insertion method to be used when and where the layer insertion method here proposed can not be profitably used. We are indeed already working in this direction, in order to expand the applicability of the present software to handle these more general cases.

The method here proposed seems well suited for the analysis of hovering rotors, with the limitations noted above. The method is ideally suited to the growth of boundary layers for Navier-Stokes simulations, both around the blades and the fuselage. For the refinement of certain internal solution features, as for example in the analysis of the acoustic pressure field for hovering rotors, as shown in figure 29, the region of localization is also well suited to the application of the present procedure. However, we remark once again that a greater flexibility would be obtained by adding to this existing capability the possibility of refining by metric-based point insertion, for example in the wake and tip-vortex regions. Future work will in fact concentrate the research in this specific direction.

Regarding the area of a-posteriori error estimation, we have reported on a generalization of the ZZ patch-recovery technique to the case of the Euler and Navier-Stokes equations. This method performs extremely well on regular meshes, but can also be used with success on the irregular meshes that characterize realistic unstructured applications. Furthermore, the method in its present implementation is very robust, cheap to compute and parallelizable with great efficiency being local in nature. In this sense, this method seems to offer a mature methodology for the estimation of the error associated with finite element solutions of complex, large scale problems. This allows to avoid the use of gradient norms for driving the adaption process, that, providing no information on the finite element error but simply tracking regions of high solution variation, can not be fully automated. We have tested with success the procedure on a number of examples, and we were able to include it in the parallel adaptive code with no particular difficulty. Furthermore, is has been experimentally observed that the method behaves well even in the presence of discontinuities, which is the typical case of Euler shocks. While the mathematical theory behind this recovery-based estimator will clearly not hold in this case, the procedure is still able to identify the regions that need to be better resolved.

Although the ZZ method has a number of useful characteristics, it was here noted that it does not provide per-se information for constructing metric fields defined on the computational domain to be used for driving anisotropic adaption processes. To overcome this limitation, we have proposed to resort to interpolation estimates that include the information on the anisotropy of

the solution as well as the anisotropy of the computational grid. With the use of such estimates, one obtains a definition of a local metric that can be used in an iterative fashion to obtain an anisotropically adapted locally optimal mesh. Unfortunately, such estimates are prohibitively expensive to compute for realistic non-linear non-self adjoint large scale applications, like the ones that are relevant to this research project. To overcome this difficulty, we have proposed to exploit the sinergy between the anisotropic estimates and the ZZ recovery technique. This way, one can compute an approximation to the local error appearing in the anisotropic estimates using the robust and cheap ZZ method, making the anisotropic information readily available for use in a fully automated adaption process. So far we have analyzed this approach from a mathematical point of view and we have numerically tested it on academic examples. The results so obtained are very promising and encourage to continue the work in this direction towards the generalization of these ideas to the compressible Euler and Navier-Stokes equations. At present we do no have results to corroborate these preliminary findings, but we intend to pursue the research in this direction in order to clarify the suitability of the aproach for applications of engineering interest.

Comparing the proposed tasks and milestone with the achieved results, we can say that we have analyzed and tested the methods here proposed on meaningful examples, and we have already implemented all the most mature procedures in the parallel-adaptive rotor code. This part of the work would have been tasks 2.1 and 2.2 of a possible second year continuation of this contract, according to the original proposal. Parts which are less mature, and in particular the anisotropic estimators, are beeing analyzed using simpler research codes.

The results of this research activity have been reported in the following scientific publications:

- Bottasso, C.L., Detomi, D. (2002) A procedure for tetrahedral boundary layer mesh generation. Engineering with Computers, accepted, to appear.
- Micheletti, S., Perotto, S. (2001) An anisotropic recovery-based a-posteriori error estimator. Proccedings of ENUMATH 2001, European Conference on Numerical Mathematics and Advanced Applications.
- Bottasso, C.L., Detomi, D. (2002) Procedures for the refinement and generation of anisotropic tetrahedral grids in three-dimensional domains. SIMAI 2002, Chia, CA, Italy, May 27–31, 2002.
- Bottasso, C.L., Detomi, D. (2002) Tetrahedral boundary layer mesh generation for viscous flows in complex geometries. XVI Congresso Nazionale AIDAA, Palermo, Italy, September 24–28, 2001.
- Bottasso, C.L. Detomi, D. (2002) Generation and refinement of anisotropic tetrahedral and hybrid grids in three dimensions. ANISGRID 2002, Politecnico di Milano, Italy, June 21, 2002.

- Formaggia, L., Micheletti, S. (2002) An anisotropic framework for error estimates in FEM. ANISGRID 2002, Politecnico di Milano, Italy, June 21, 2002.
- Bottasso, C.L., Detomi, D. (2002) Insertion of anisotropic layers in three-dimensional tetrahedral grids. In preparation for submission.

Finally, we remark that in June 2002 we have organized a one-day workshop at the Politecnico di Milano on the topic of anisotropic grids:

> ANISGRID 2002. *Anisotropic Grids: Generation, Adaption and Error Estimation.* Organizers: C.L. Bottasso, S. Micheletti, S. Perotto, R. Sacco. Politecnico di Milano, Milano, Italy, June 21, 2002.

Speakers and participants were from the European academic and industrial worlds. This initiative, although not directly financed by this research contract, was made possible by the research activity described herein and was directly inspired by it.

REFERENCES

[1] Ainsworth, M., Oden, J.T. (2001) A posteriori error estimation in finite element analysis. John Wiley & Sons, Inc.

[2] Babuska, I., Aziz, A.K. (1976) On the angle condition in the finite element method. SIAM Journal on Numerical Analysis, 13, 214–226.

[3] Batina, J.T. (1989) Unsteady Euler airfoil solution using unstructured dynamic meshes. AIAA Paper No. 89–0115, AIAA 27th Aerospace Sciences Meeting, Reno, Nevada, USA.

[4] Beall, M.W., Shephard, M.S. (1997) A general topology-based mesh data structure. International Journal for Numerical Methods in Engineering, 40, 1573–1596.

[5] Borouchaki, H., George, P.L. (1997) Delaunay mesh generation governed by metric specification. Part I. Algorithms. Finite Elements in Analysis and Design, 25, 85–109.

[6] Bottasso, C.L., Detomi, D. (2002) A procedure for tetrahedral boundary layer mesh generation. Engineering with Computers, accepted, to appear.

[7] Bottasso, C.L., Detomi, D. (2002) Insertion of anisotropic layers in three-dimensional tetrahedral grids. In preparation for submission.

[8] Bottasso, C.L., Shephard, M.S. (2000) Finite element adaptive multigrid euler solver for rotary wing aerodynamics. AIAA Journal, 38, 50–56.

[9] Bottasso, C.L., Shephard, M.S. (1997) A parallel adaptive finite element euler flow solver for rotary wing aerodynamics. AIAA Journal, 35, 937–944.

[10] Connell, S.D., Braaten, M.E. (1995) Semistructured mesh generation for three-dimensional Navier-Stokes calculation. AIAA Journal, 33, 1017–1024.

[11] De Cougny, H.L., Shephard, M.S. (1995) Parallel mesh adaptation by local mesh modification. Scientific Report 21–95, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY.

[12] Farhat, C., Degand, C., Koobus, B., Lesoinne, M. (1998) An improved method of spring analogy for dynamic unstructured fluid meshes. AIAA Paper No. 98–2070, AIAA 39th Structures, Structural Dynamics, and Materials Conference, Long Beach, CA, USA.

[13] Farhat, C., Lesoinne, M. (1993) Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. International Journal for Numerical Methods in Engineering, 36, 745–764.

[14] Field, D.A. (1988) Laplacian smoothing and Delaunay triangulations. Communications in Numerical Methods in Engineering, 4, 709–712.

[15] Garimella, R.V., Shephard, M.S. (2000) Boundary layer mesh generation for viscous flow simulations. International Journal for Numerical Methods in Engineering, 49, 193–218.

[16] George, P.L., Borouchaki, H. (1998) Delaunay triangulation and meshing – Application to finite elements. Editions Hermes, Paris.

[17] Kallinderis, Y., Khawaja, A., Mc Morris, H. (1996) Hybrid prismatic tetrahedral grid generation for viscous flows around complex geometries. AIAA Journal, 34, 291–298.

[18] Kallinderis, Y., Ward, S. (1993) Prismatic grid generation for three-dimensional complex geometries. AIAA Journal, 31, 1850–1856.

[19] Krízek, M., Neittaanmäki, P. (1987) On a global superconvergence of the gradient of linear triangular elements. Journal of Computational and Applied Mathematics, 18, 221-233.

[20] Li, X., Shephard, M.S., Beall, M.W. (2001) Accounting for curved domains in mesh refinement. VI US National Congress on Computational Mechanics, Dearborn, MI, USA.

[21] Löner, R. (1995) Matching semi-structured and unstructured grids for Navier-Stokes computations. AIAA-93-3348-CP.

[22] Micheletti, S., Perotto, S. (2001) An anisotropic recovery-based a-posteriori error estimator. Proccedings of ENUMATH 2001, European Conference on Numerical Mathematics and Advanced Applications.

[23] Mc Morris, H., Kallinderis, Y. (1997) Octree-advancing front method for generation of unstructured surface and volume meshes. AIAA Journal, 35, 976–984.

[24] Rodriguez, R. (1994) Some remarks on the Zienkiewicz-Zhu estimator. Numerical Methods for Partial Differential Equations, 10, 625–635.

[25] Schroeder, W.J., Shephard, M.S. (1991) On rigorous conditions for automatically generated finite element meshes. In Turner, J., Pegna, J., Wozny, M., editors, Product Modeling for Computer Aided Design and Manufacturing, North-Holland, 267–281.

[26] Shakib, F., Hughes, T.J.R., Johan, Z. (1991) A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier Stokes Equations. Computer Methods in Applied Mechanics and Engineering, 89, 141-219.

[27] Verfürth, R. (1996) A review of a posteriori error estimation and adaptive mesh-refinement techniques. Wiley-Teubner, Chichester-Stuttgart.

[28] Zienkiewicz, O.C., Wu, J. (1994) Automatic directional refinement in adaptive analysis of compressible flows. International Journal for Numerical Methods in Engineering, 37, 2189–2210.

[29] Zienkiewicz, O.C., Zhu, J.Z. (1992) The superconvergent patch recovery and a-posteriori error estimates. Part I: the recovery technique. International Journal for Numerical Methods in Engineering, 33, 1331–1364.

[30] Zienkiewicz, O.C., Zhu, J.Z. (1992) The superconvergent patch recovery and a-posteriori error estimates. Part II: error estimates and adaptivity. International Journal for Numerical Methods in Engineering, 33, 1365–1382.

FIGURES

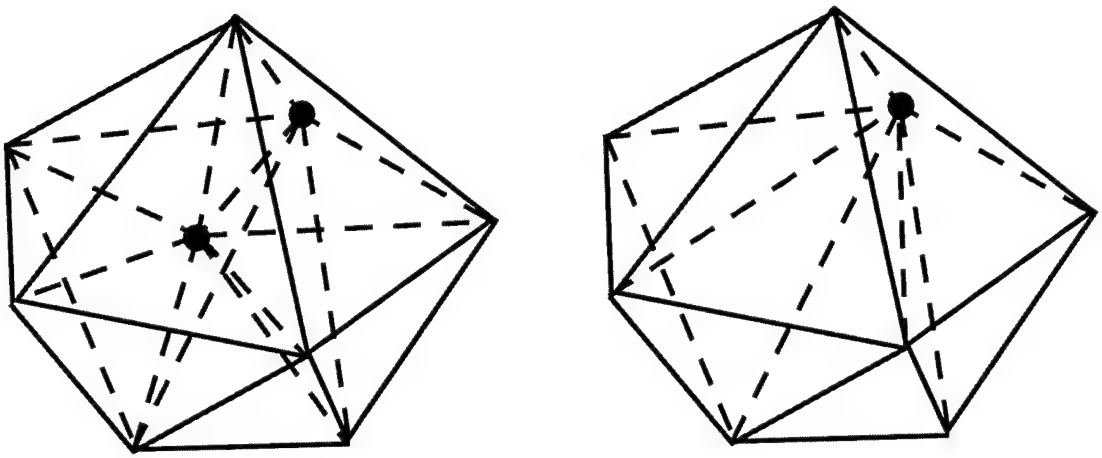

FIGURE 1. Edge collapsing in three dimensions.

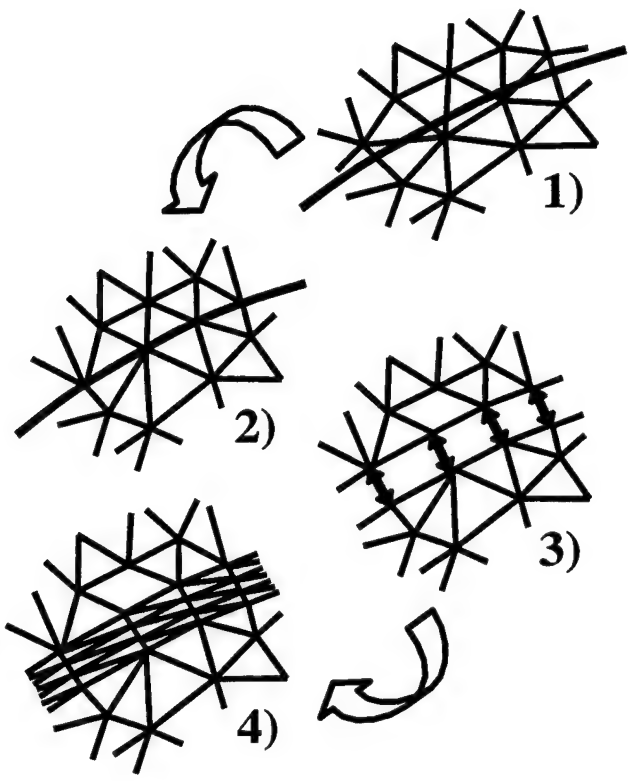

FIGURE 2. Anisotropic adaption by mesh cutting, opening and filling. Top: original mesh and trace of the surface. Middle: mesh after the cut and the local optimization. Bottom: final adapted grid.

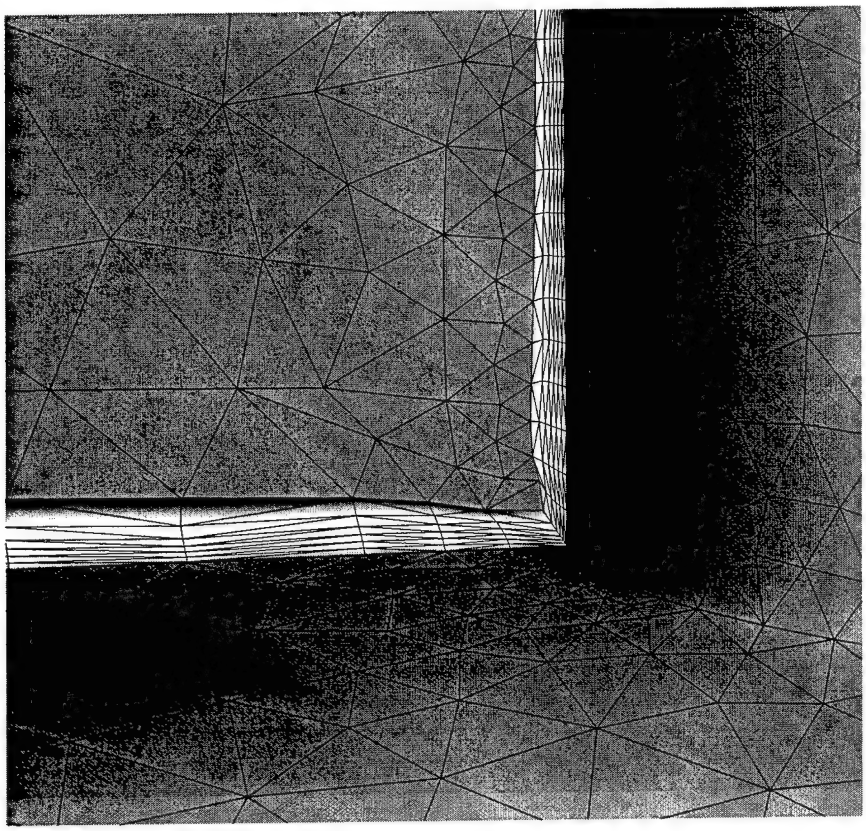

FIGURE 3. Automatic curving of growth curves through mesh relaxation in the boundary layer region.

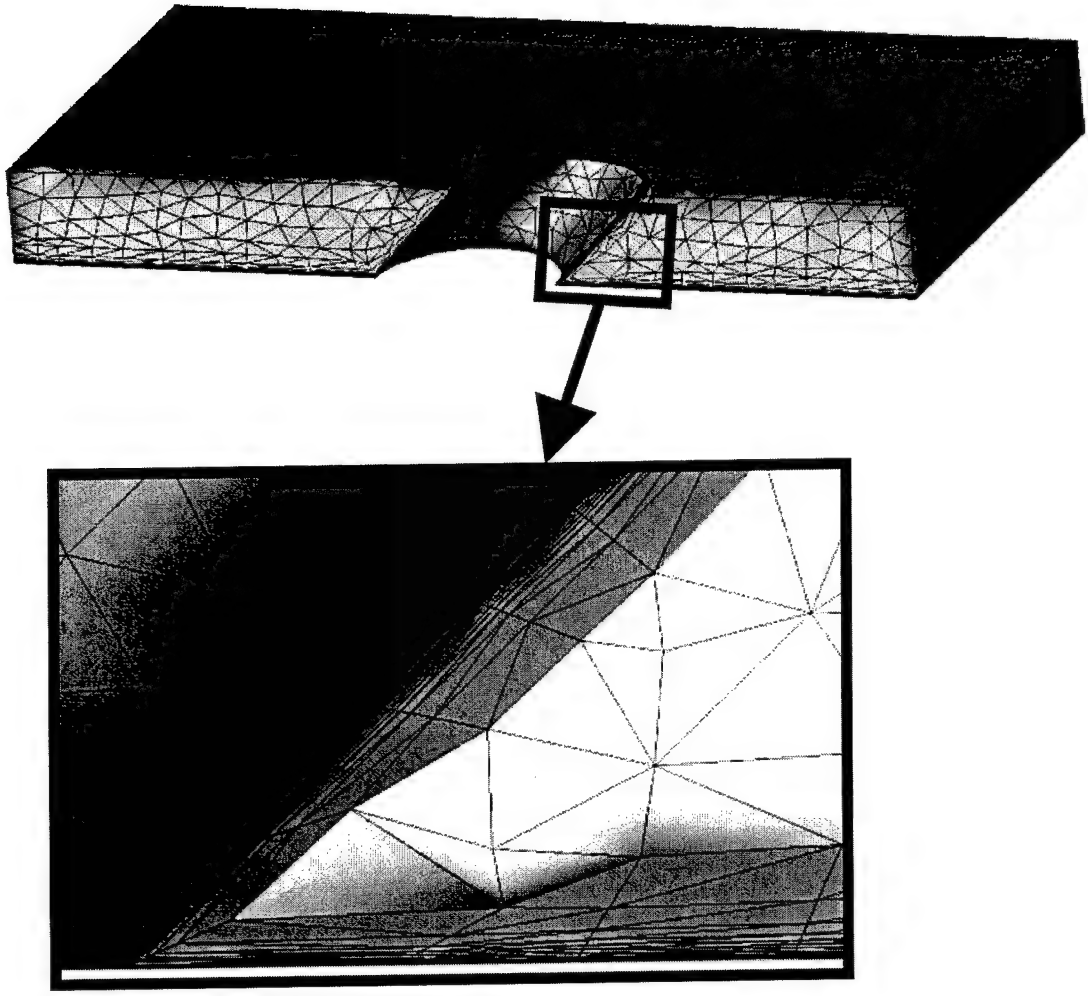

FIGURE 4. Growth of a boundary layer mesh in a sharp corner region.

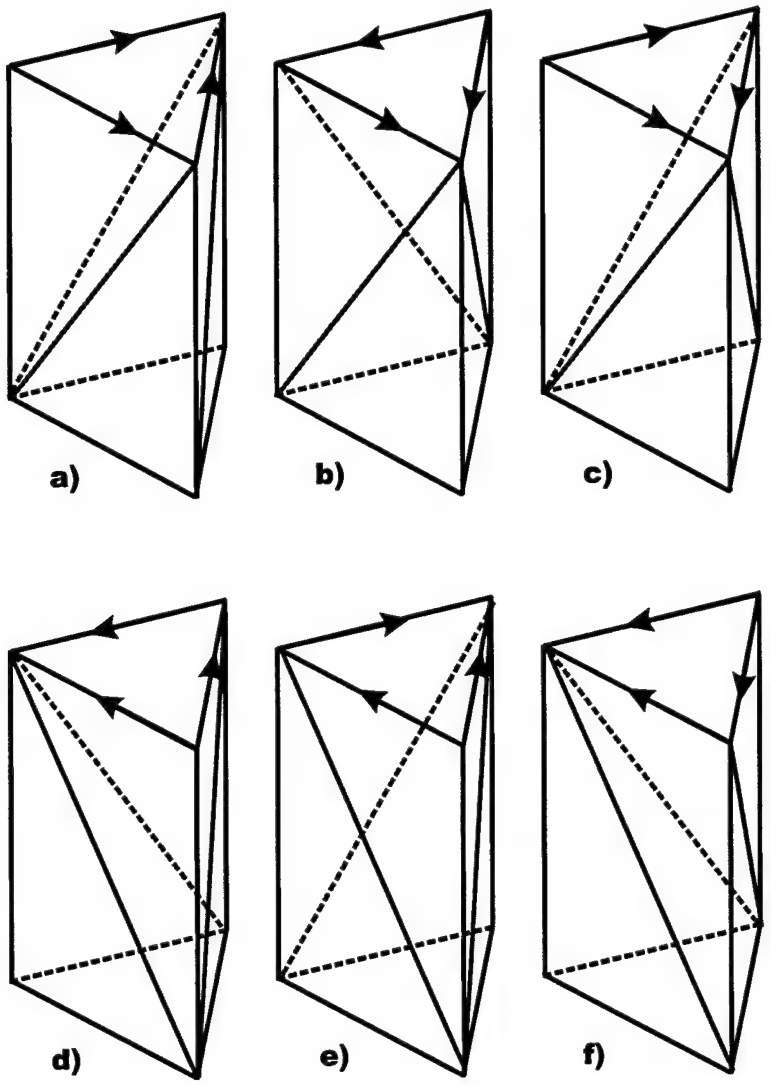

FIGURE 5. Prism splitting templates and their symbolical representation through the orientation of face edges.

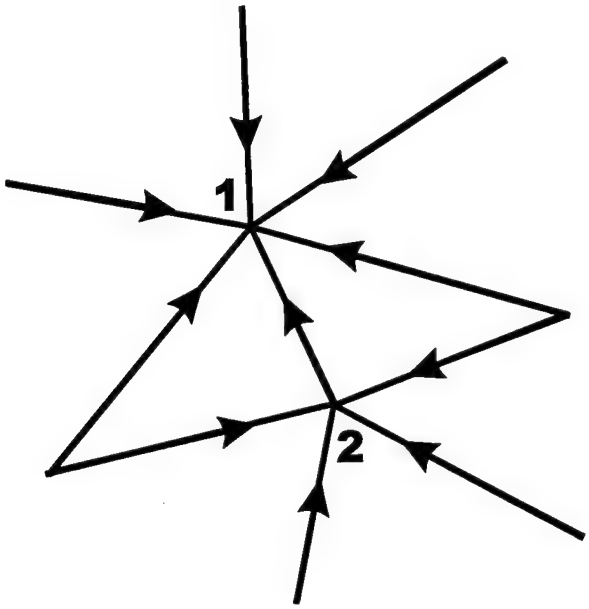

FIGURE 6. Consistent prism splitting algorithm.

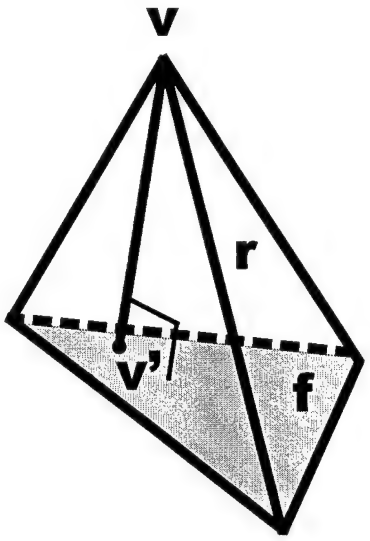

FIGURE 7. Additional linear springs connecting each vertex with the opposite face in a tetrahedron.

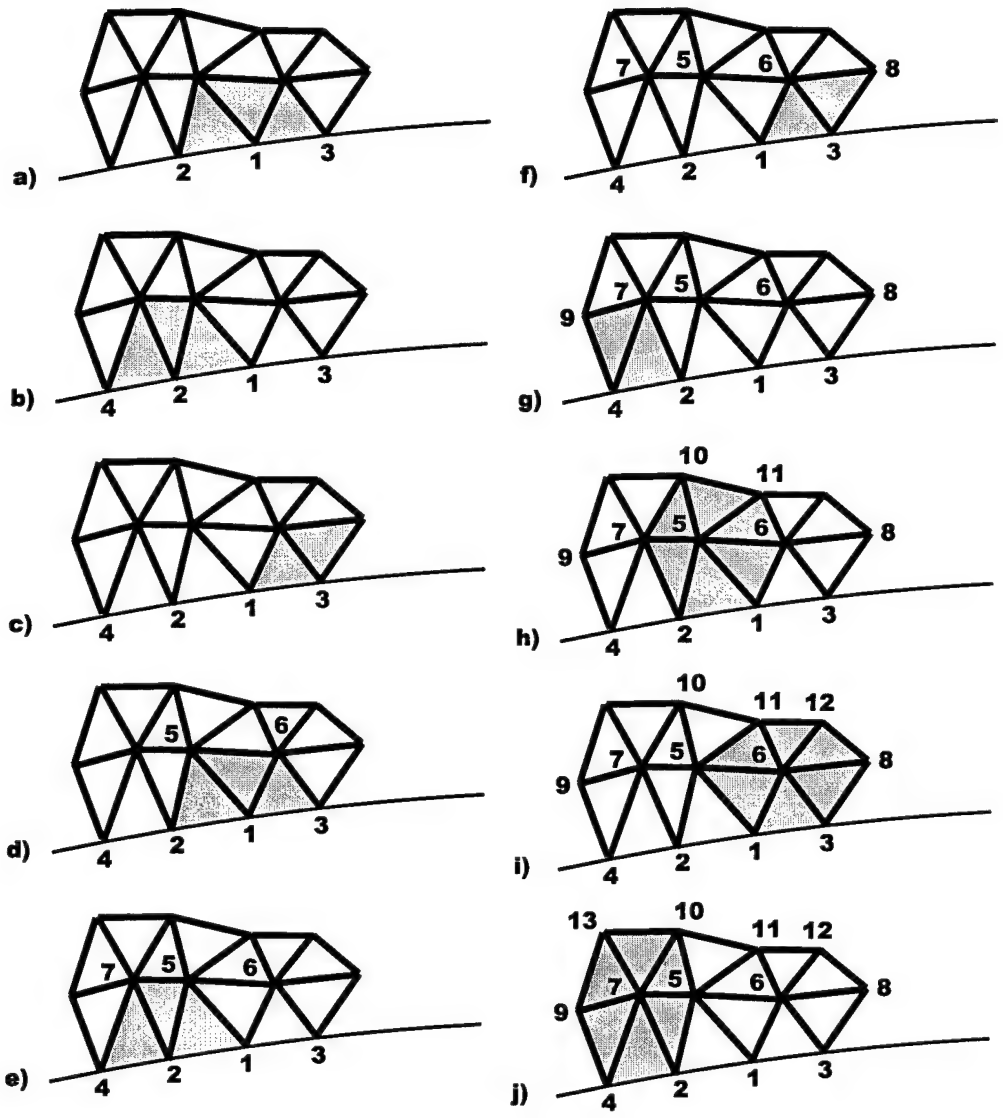

FIGURE 8. Construction of a line ordering in an unstructured mesh using the "onion leaves" greedy algorithm.

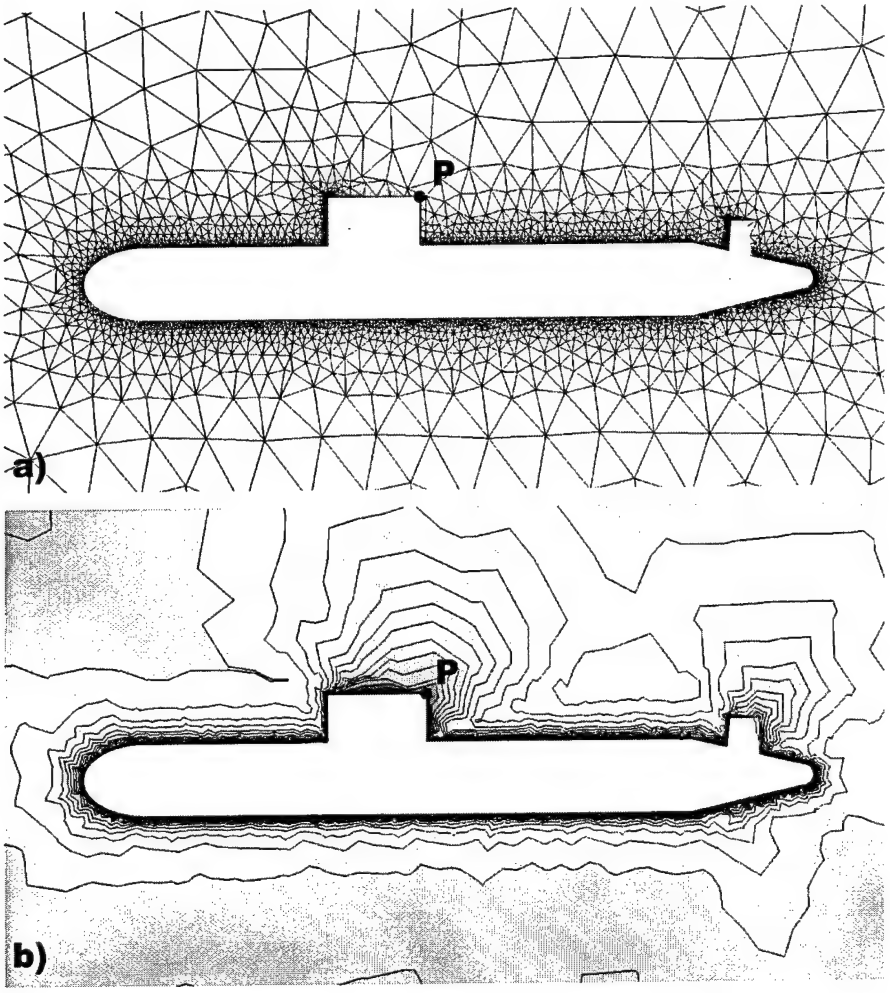

FIGURE 9. Ordering for the mesh of a two-dimensional submarine model.

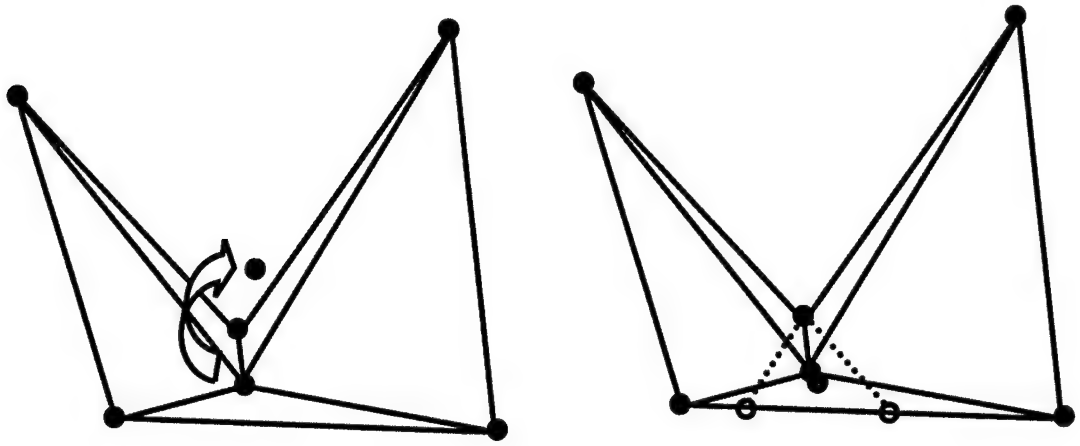

FIGURE 10. At left, centroid of concave region yields invalid mesh. At right, centroid of "regularized" region yields valid mesh.

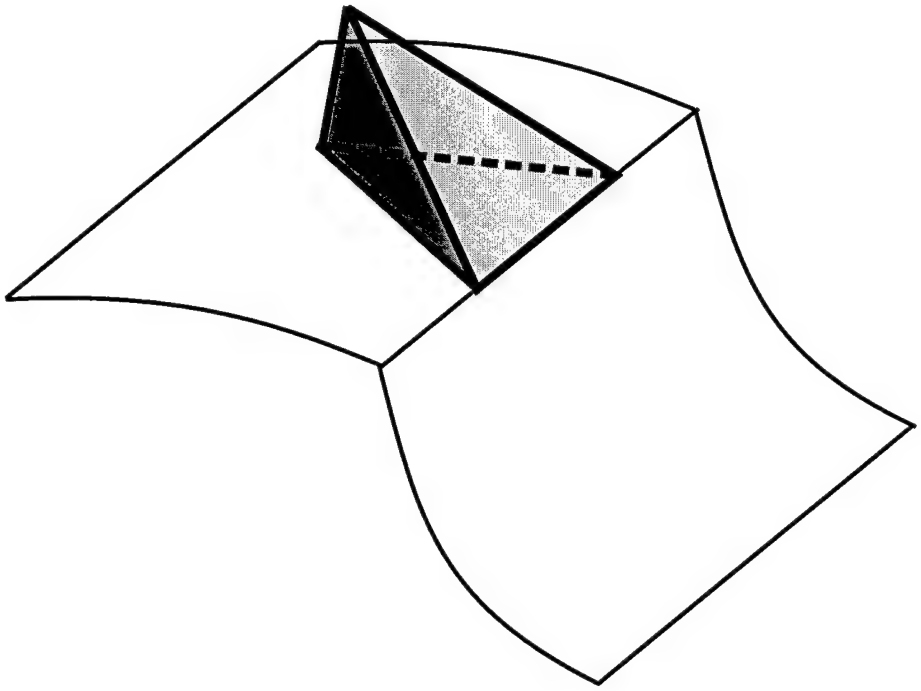

FIGURE 11. Boundary layer ending at a model edge.

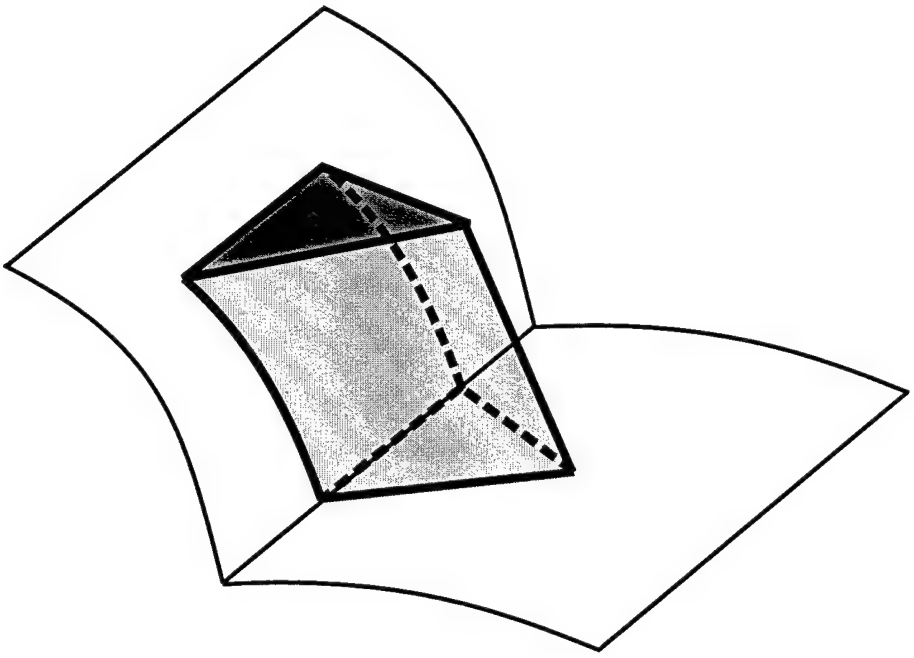

FIGURE 12. Boundary layer ending at a model face.

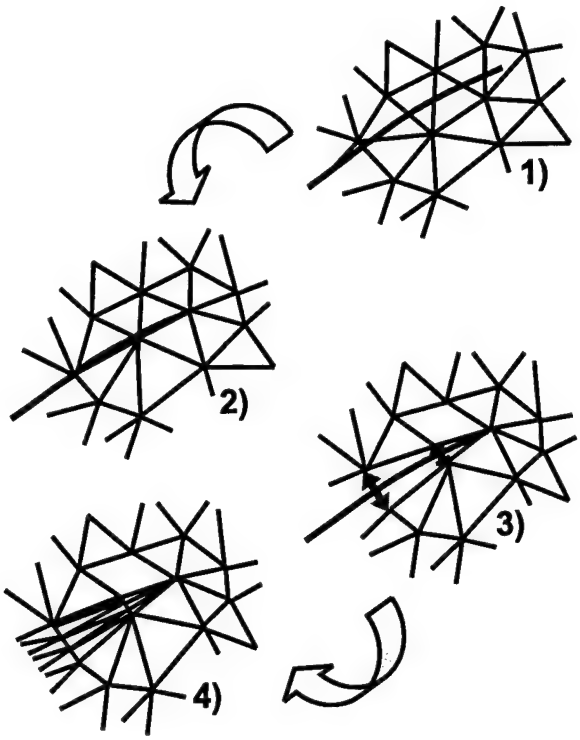

FIGURE 13. Internal layer ending within the domain, i.e. without intersecting a model boundary.

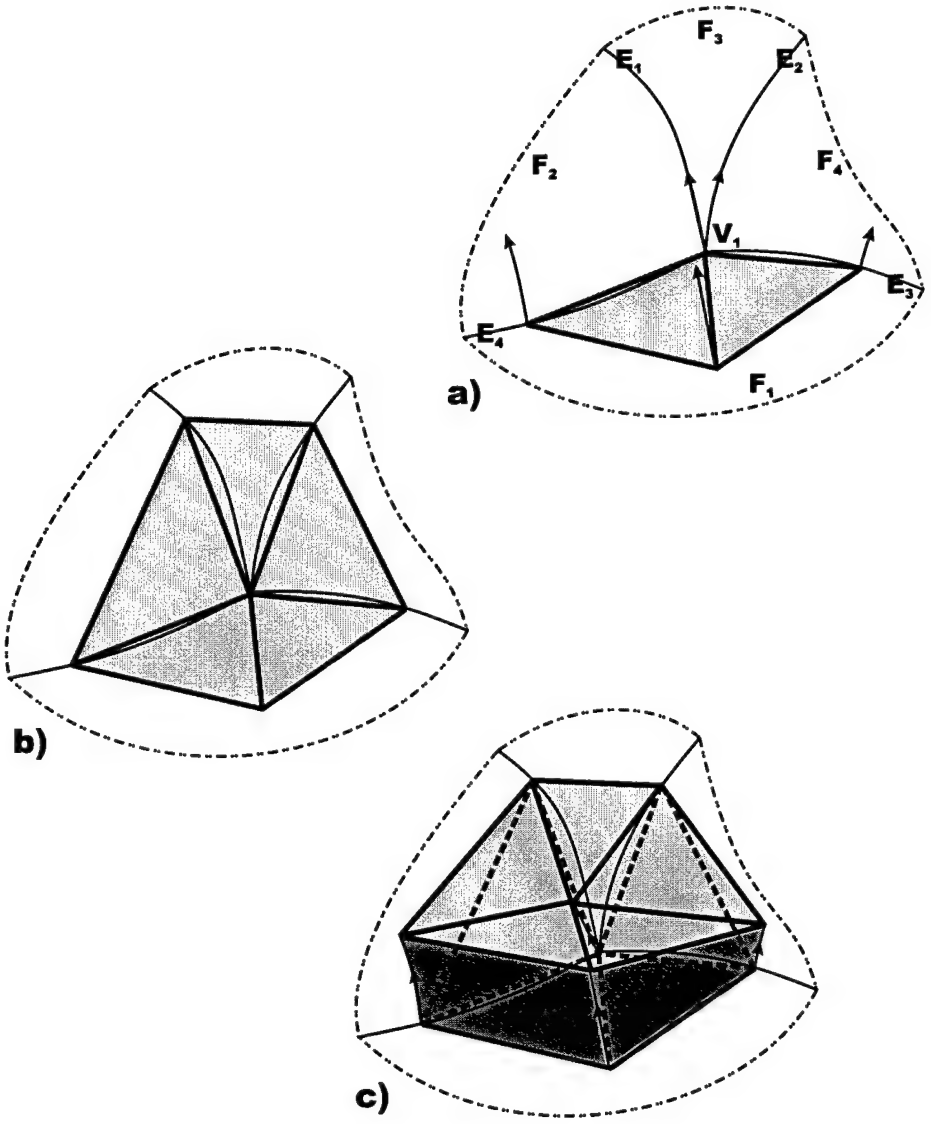

FIGURE 14. Boundary layer spillover. a) Required growth directions without spillover; b) patch of mesh faces that will be detached from the model; c) detachment and creation of boundary layer with spillover.

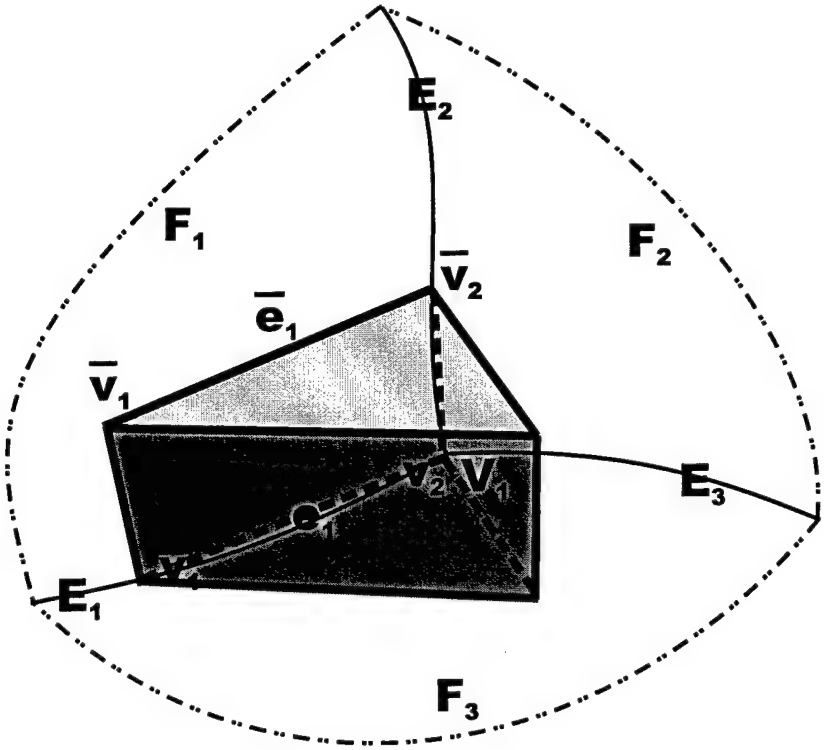

FIGURE 15. Classification against model entities of the newly created mesh entities.

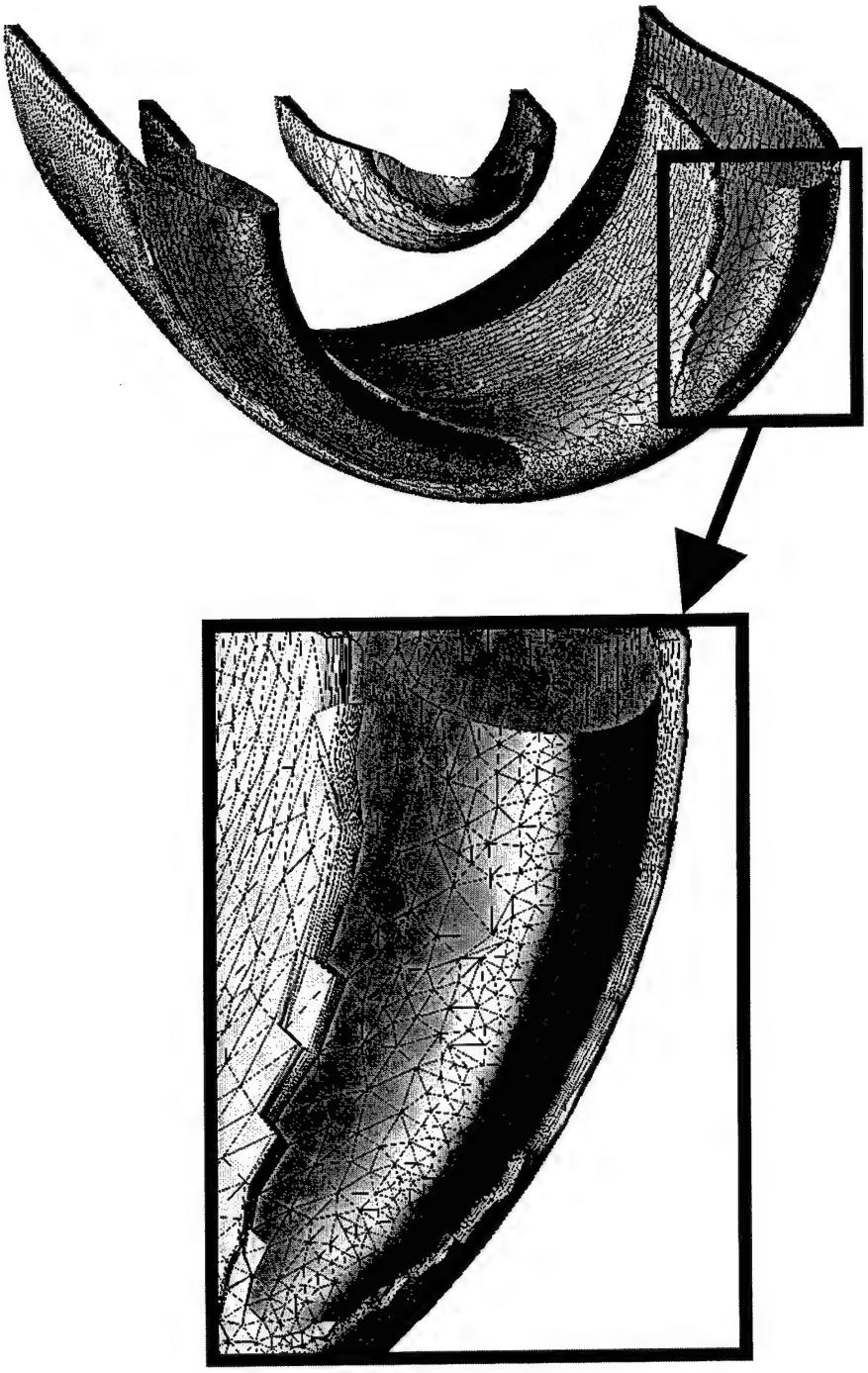

FIGURE 16. Boundary layer mesh grown around an engine nacelle model. Parts of the anisotropic grid were deleted for exposing the through-the-thickness discretization and the original model face triangulation.

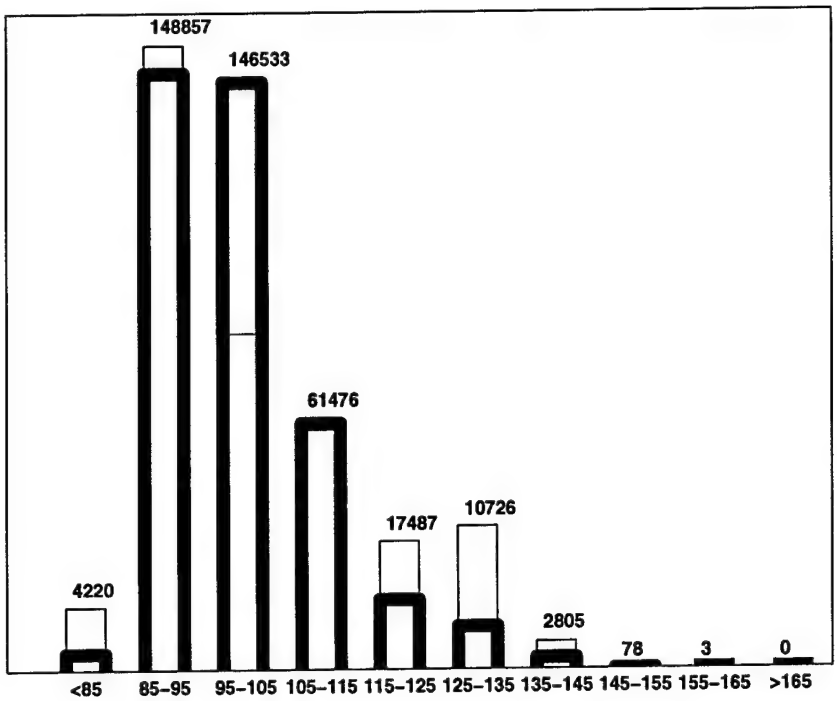

FIGURE 17. Histogram of the maximum dihedral angles for each element of the initial (thin lines) and final (thick lines) meshes of the nacelle model.

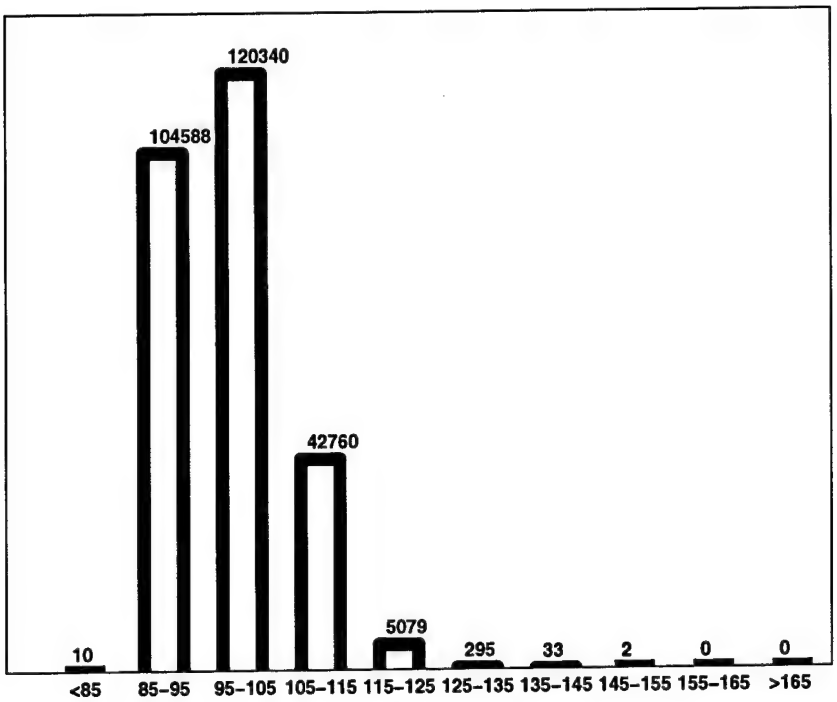

FIGURE 18. Histogram of the maximum dihedral angles for the anisotropic part of the adapted grid.

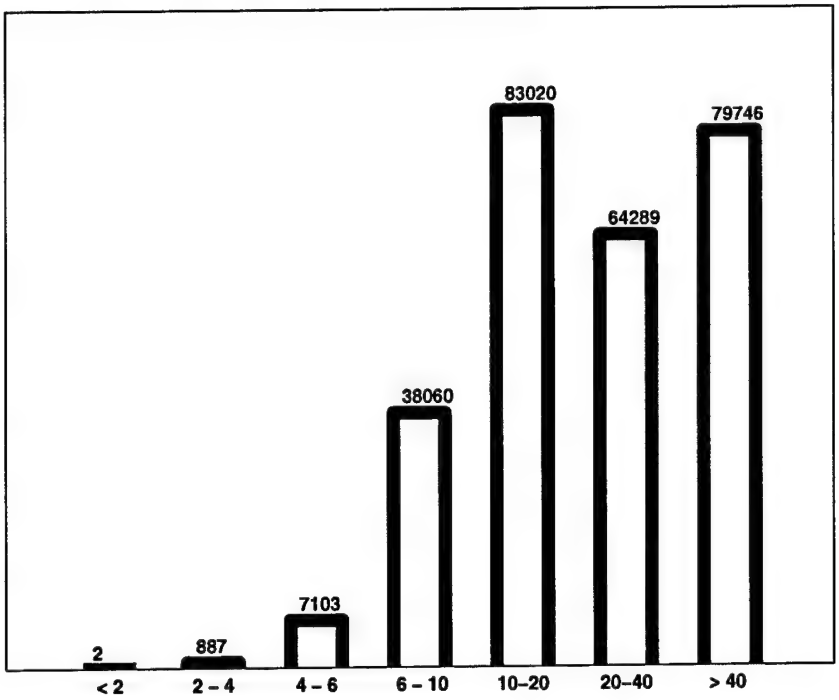

FIGURE 19. Histogram of the ratio of the radii of the inscribed and circumscribed spheres for the anisotropic part of the adapted grid.

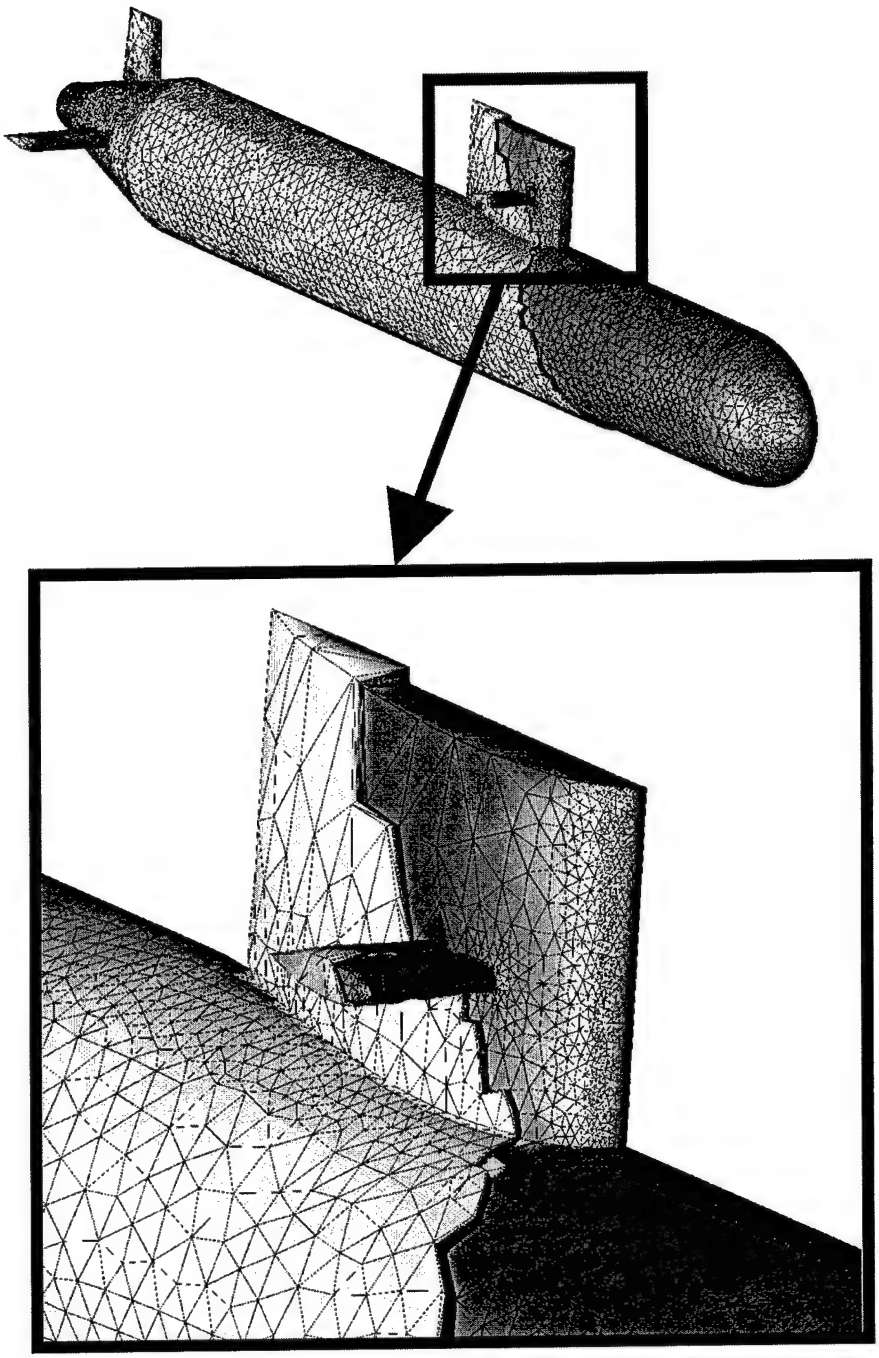

FIGURE 20. Boundary layer mesh grown around a submarine model. Parts of the anisotropic grid were deleted for exposing the through-the-thickness discretization and the original model face triangulation.

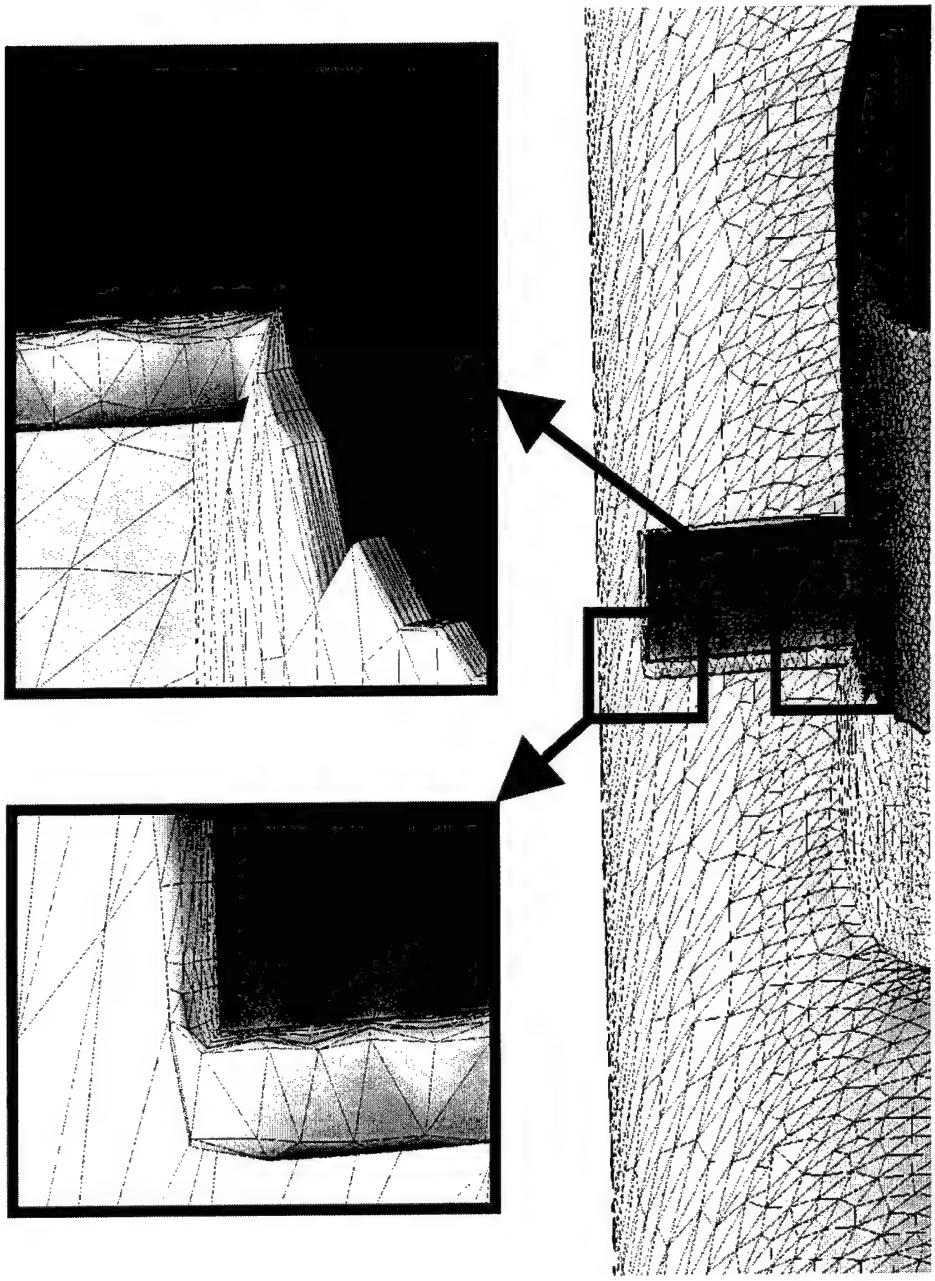

FIGURE 21. Details of a boundary layer mesh grown around a submarine model.

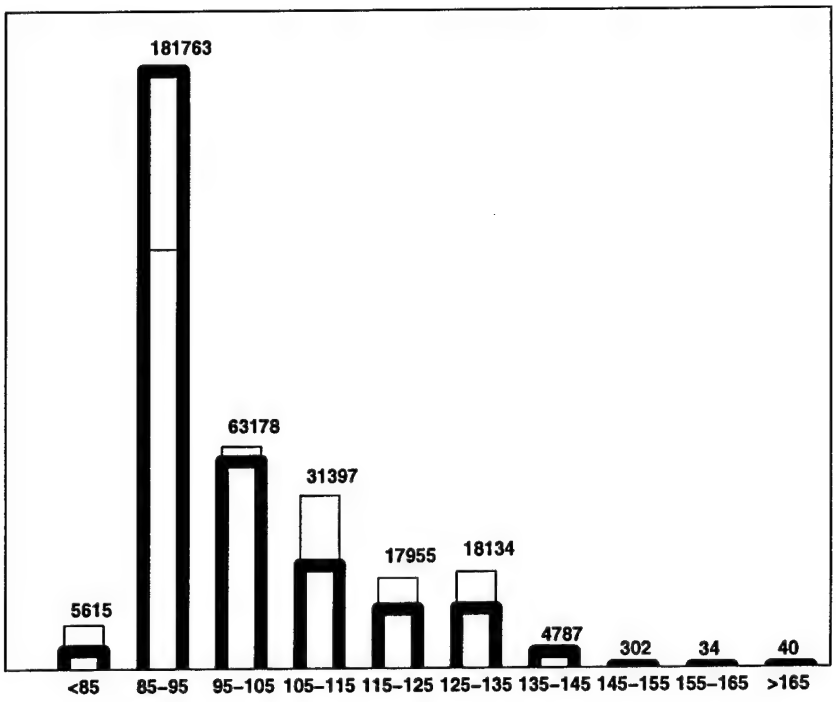

FIGURE 22. Histogram of the maximum dihedral angles for each element of the initial (thin lines) and final (thick lines) meshes of the submarine model.

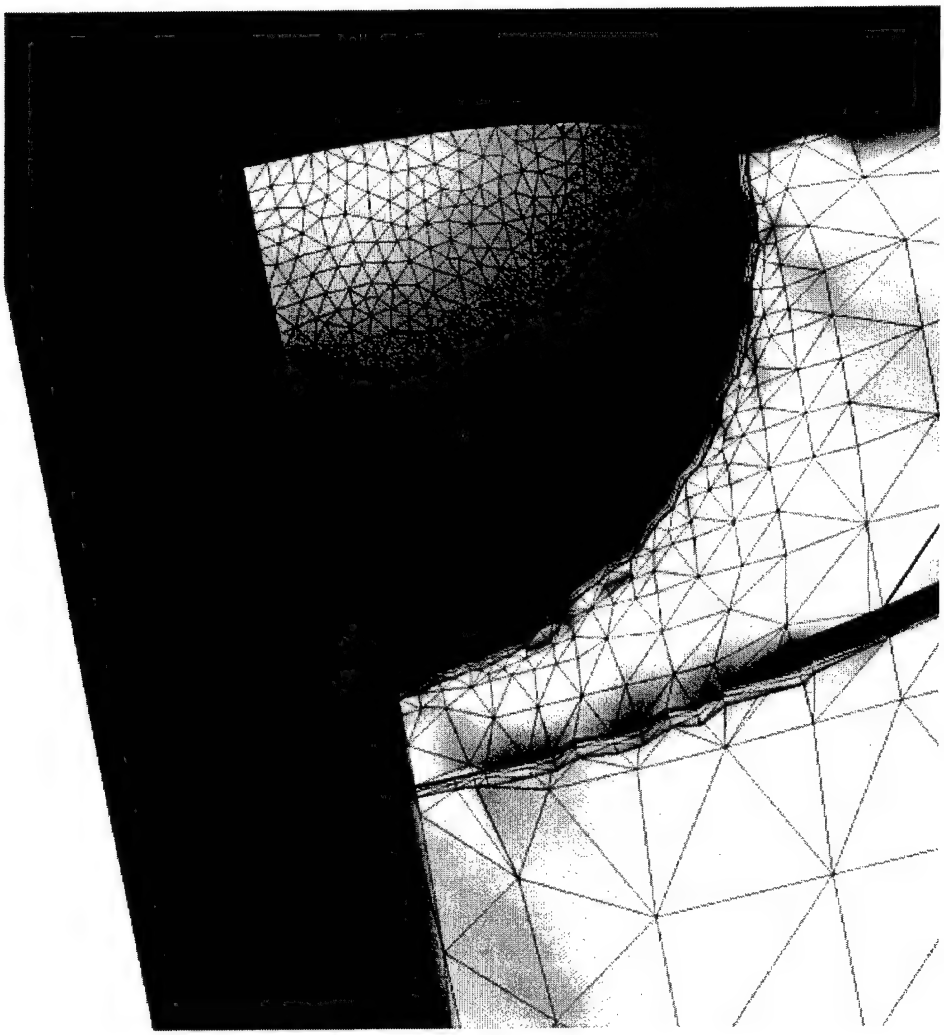

FIGURE 23. Insertion of a thin internal layer in front of a sphere.

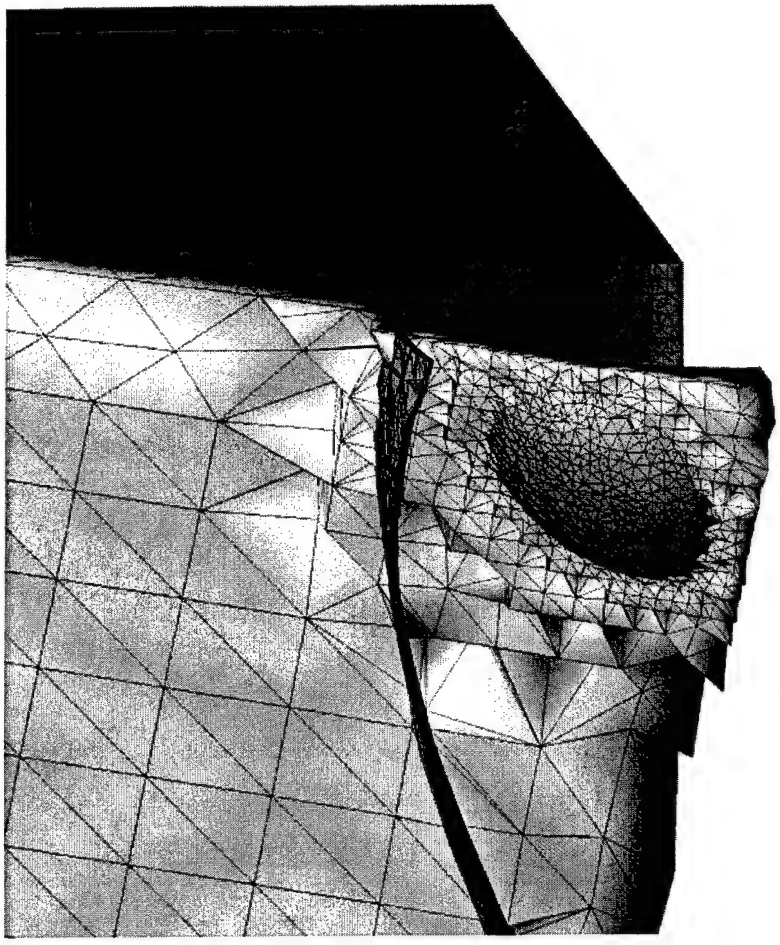

FIGURE 24. Insertion of a thin internal layer in front of a sphere. Elements in the isotropic grid were stripped away in order to show the anisotropic mesh.

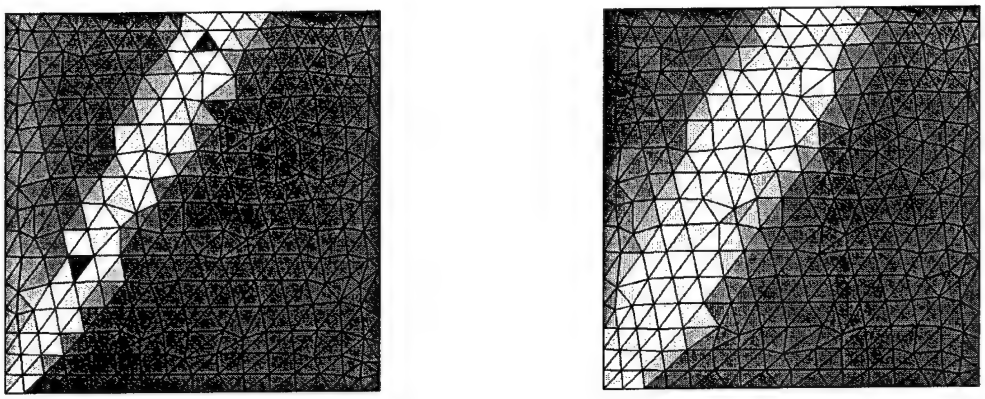

FIGURE 25. True (left) and ZZ estimated (right) errors for the oblique shock problem.

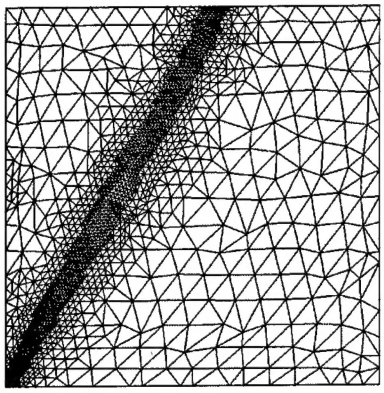

FIGURE 26. Adapted grid after two refinements for the oblique shock problem.

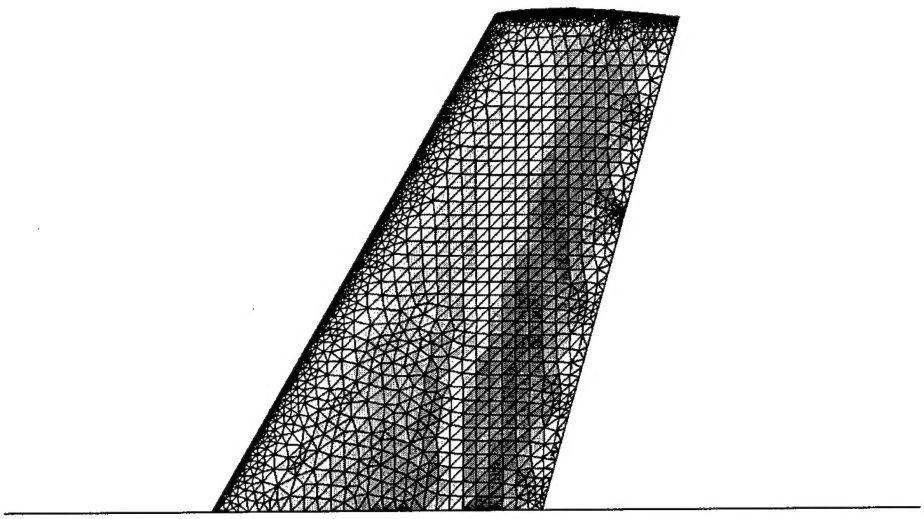

FIGURE 27. ZZ estimated error for the Onera M6 wing problem.

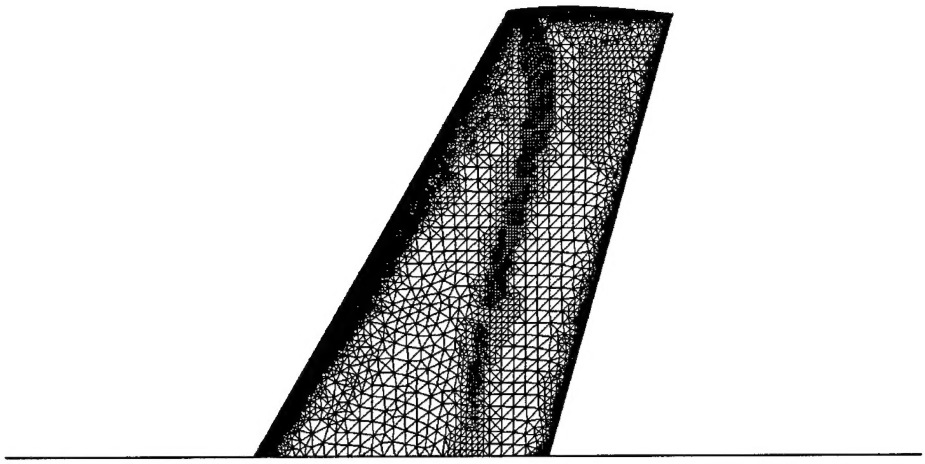

FIGURE 28. Adapted grid after two refinements for the Onera M6 wing problem.

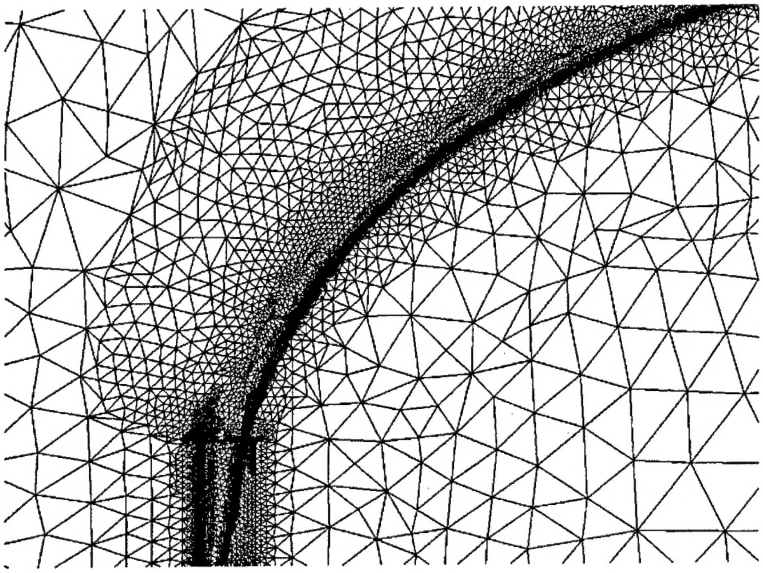

FIGURE 29. View of an isotropically refined mesh for a high speed impusive noise computation.